



L'algoritmo del banchiere Implementazione in REALbasic

Documentazione

Autore

eldino

<http://eldino.wordpress.com>
numbworks@gmail.com

La teoria

L'algoritmo del banchiere è un algoritmo finalizzato ad evitare le situazioni di stallo dei processi quando il sistema prevede l'assegnazione delle risorse con più istanze per ciascun tipo di risorsa. L'algoritmo del banchiere è di fatto costituito da due sub-algoritmi:

- **Algoritmo di richiesta delle risorse**
- **Algoritmo di verifica della sicurezza**

L'**algoritmo di richiesta delle risorse** effettua una serie di verifiche sulla richiesta effettuata dal processo e si articola in tre steps:

STEP1

$Richieste(i) \leq Necessità(i)$: la richiesta effettuata dal processo deve essere minore o uguale della necessità residua di risorse del processo stesso; se questa condizione si verifica, si procede con lo step2, altrimenti si riporta una condizione d'errore.

STEP2

$Richieste(i) \leq Disponibili$: la richiesta effettuata dal processo deve essere compatibile con il numero delle istanze disponibili per ciascun tipo di risorsa; se questa condizione si verifica, si procede con lo step3, altrimenti si riporta una condizione d'errore.

STEP3: il sistema simula l'assegnazione delle risorse richieste al processo $P(i)$, modificando il sistema nel seguente modo:

$$\begin{aligned} Disponibili &= Disponibili - Richieste(i) \\ Assegnate(i) &= Assegnate(i) + Richieste(i) \\ Necessità(i) &= Necessità(i) - Richieste(i) \end{aligned}$$

Se il nuovo sistema si rivela essere in uno stato sicuro, allora la transazione può dirsi completa ed il nuovo sistema si mette in attesa di un'altra richiesta; se invece il nuovo sistema non si trova in uno stato sicuro, si ritorna allo stato precedente (rollback).

Per verificare che un sistema si trovi in uno stato sicuro, si utilizza l'**algoritmo di verifica della sicurezza**, che è così descritto:

STEP1

Sia m il numero di tipi di risorsa e sia n il numero di processi, e siano Lavoro e Fine vettori di lunghezza rispettivamente m e n . Inizializziamo:

$$\begin{aligned} Lavoro &:= Disponibili \\ Fine[i] &:= \text{falso} \text{ (per } 1 \dots n) \end{aligned}$$

STEP2

Cerca un indice i tale che valgano contemporaneamente le seguenti relazioni:

- a) Fine[i] falso
- b) Necessità[i] <= Lavoro

se tale i non esiste, vai allo STEP4, altrimenti vai allo STEP3.

STEP3

Lavoro := Lavoro + Assegnate[i]

Fine[i] := vero

torna allo STEP2.

STEP4

Se Fine [i] = Vero per ogni i, allora il sistema è in uno stato sicuro.

Entrambi gli algoritmi vertono su alcune strutture dati che devono essere fornite dall'utente. Queste strutture dati sono:

numero dei processi del sistema (n);

numero dei tipi di risorsa (m);

disponibili (vettore di lunghezza m che indica il numero di istanze disponibili per ciascun tipo di risorsa);

massimo (matrice n x m che contiene la richiesta massima di ciascun processo);

assegnate (matrice n x m che definisce il numero delle istanze di ciascun tipo di risorsa attualmente assegnate ad ogni processo);

necessità (matrice n x m indica la necessità residua di risorse relativa ad ogni processo);

richieste (vettore delle richieste per il processo P).

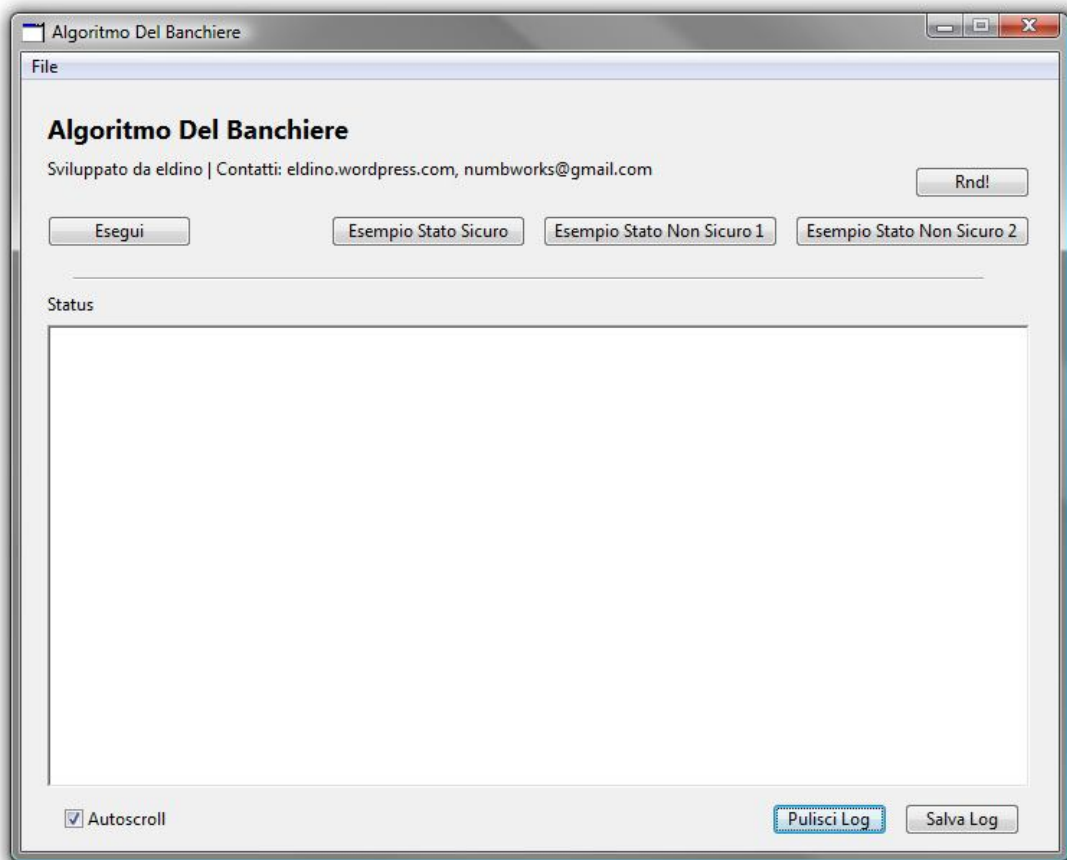
L'implementazione

Innanzitutto, la mia implementazione dell'algoritmo del banchiere è esclusivamente basata su interfaccia grafica, e da questa scelta deriva l'idea di utilizzare un linguaggio di programmazione, il REALbasic, che è molto valido in questo segmento. L'interfaccia grafica ruota tutta intorno ad una finestra principale che contiene i pulsanti per accedere alle core features del programma e ad alcune funzioni avanzate. Questa finestra funge sia da "plancia di controllo" che da console: integra, infatti, una sezione "Status" in cui il software logga tutte le operazioni man mano che le compie (sarà utilissimo per mostrare tutti gli steps dell'algoritmo di richiesta delle risorse e dell'algoritmo di verifica della sicurezza).

La scelta di posizionare una vera e propria console all'interno della finestra principale deriva da una duplice esigenza:

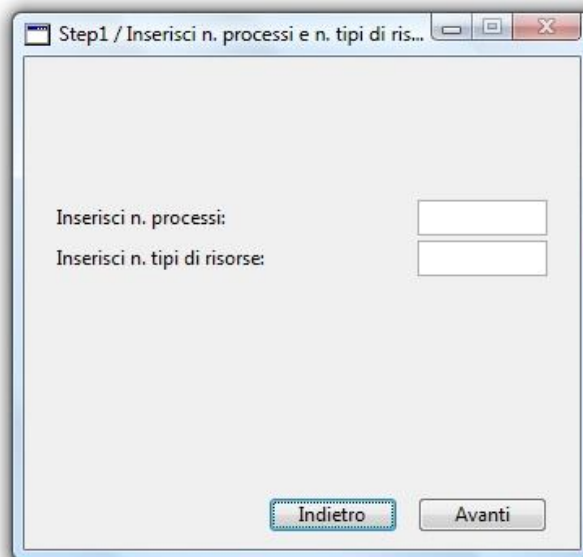
- praticità per l'utente, che può controllare man mano ciò che succede ed il frutto delle sue scelte;
- maggiore usabilità rispetto all'uso di finestrelle MsgBox et similia.

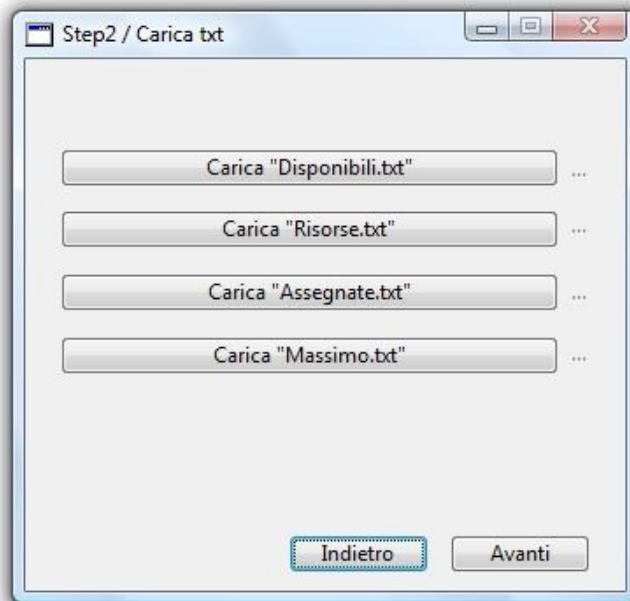
La sezione "Status" è liberamente ispirata alla console dell'IDE Eclipse.



Come si vede dallo screenshot, nella finestra principale del programma vi sono diversi pulsanti su cui l'utente può cliccare. Descriviamoli brevemente.

Il cuore del programma è il pulsante "Esegui" che permette di eseguire l'algoritmo del banchiere sui dati forniti dall'utente. Cliccando su questo pulsante si attiva un wizard in quattro step che guida l'utente nell'immissione dei dati. Questi vengono inseriti nel programma sia da tastiera che da file CSV.

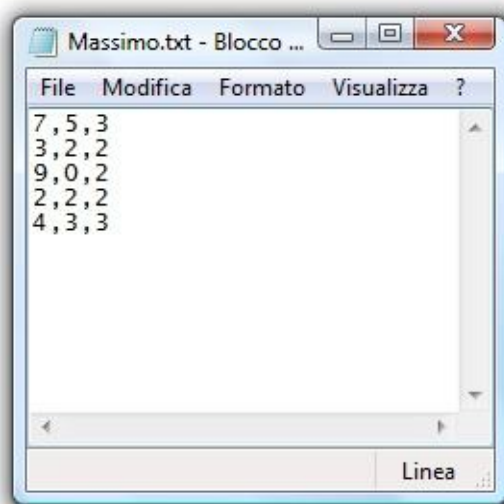
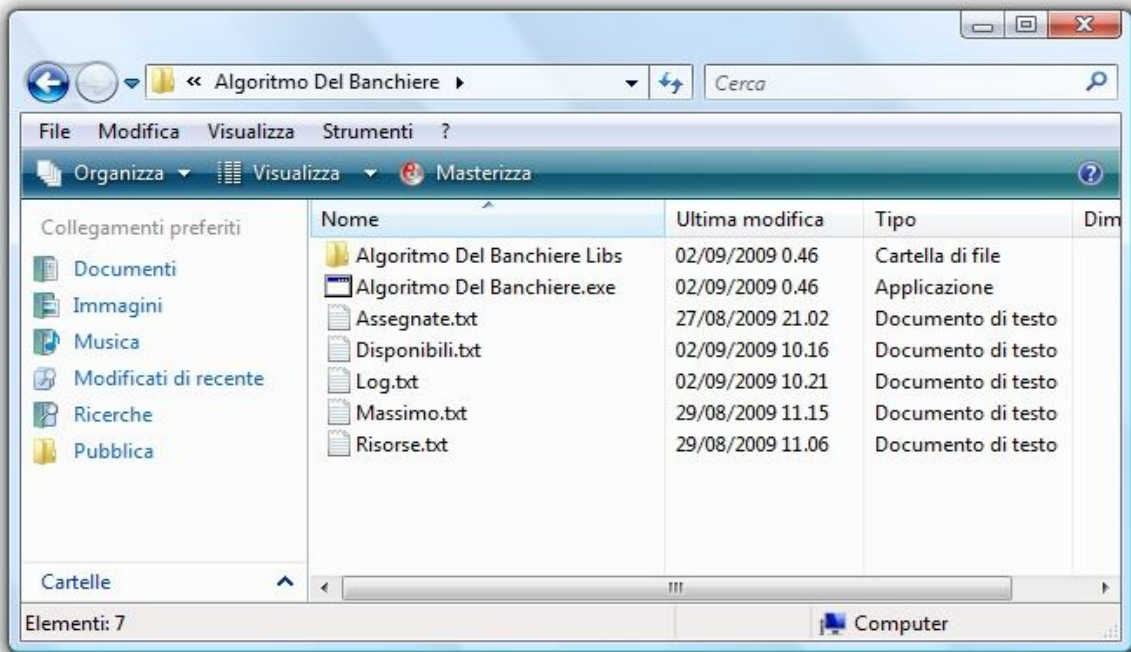




I dati più semplici da inserire (numero di processi, il numero di risorse, richiesta e processo richiedente) vengono immessi da tastiera, mentre i dati più complessi vengono caricati da alcuni specifici file di testo presenti nella cartella del software. Questi file sono: Disponibili.txt, Assegnate.txt, Risorse.txt e Massimo.txt, e sono formattati in maniera human-readable.

In particolare, le matrici vengono salvate nella forma:

0,1,2
3,4,5
6,7,8
...



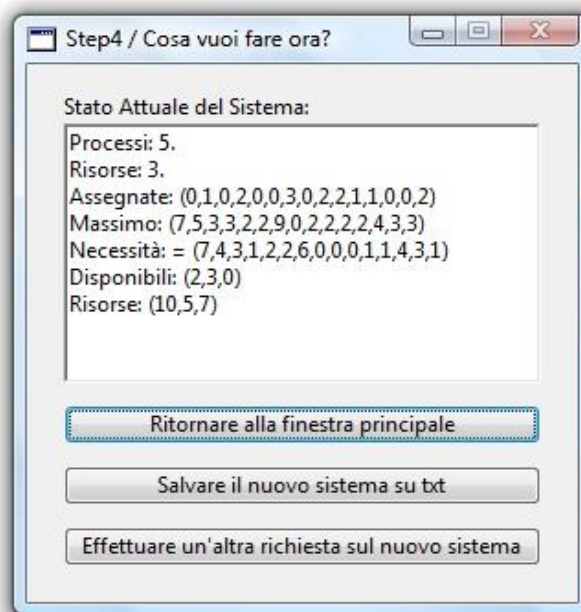
Sarà poi cura del software effettuare il parsing corretto in base ai dati inseriti da tastiera dall'utente. Ovviamente, ciò che l'utente digita deve corrispondere al contenuto dei file di testo, altrimenti il wizard non proseguirà.

Ad esempio, se si inserisce numero di processi = 5 e numero di risorse = 3, i file Assegnate.txt e Massimo.txt dovranno contenere delle matrici 5 x 3.

Ad ogni modo, in caso di mancato caricamento di un file CSV nello step 2 del wizard, l'errore sarà notificato all'utente nello status, che potrà provvedere ad editare il file incriminato e riprovare, altrimenti non potrà proseguire.

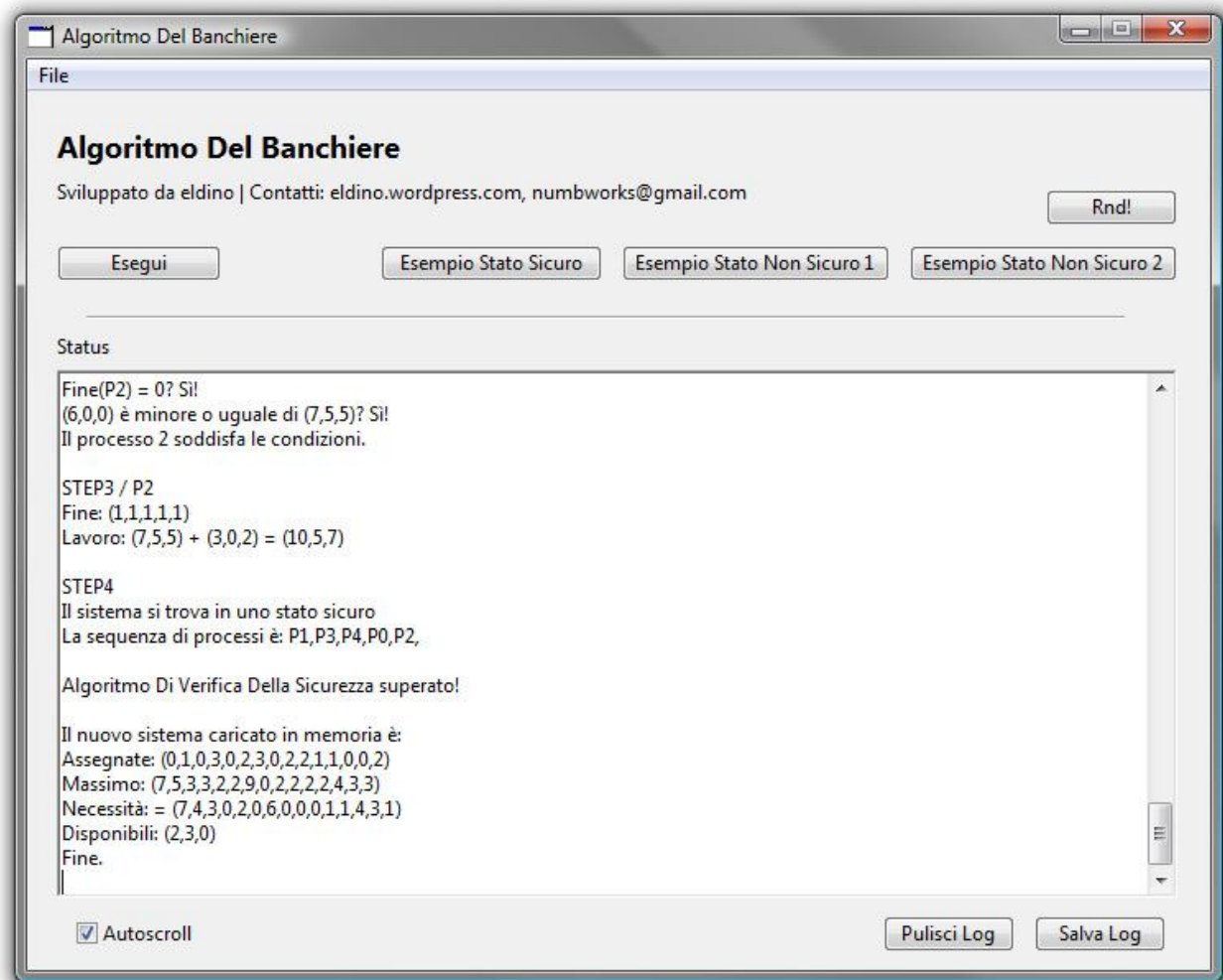
Se nei primi tre step viene inserito tutto correttamente, il software eseguirà prima l'algoritmo di richiesta delle risorse, ed in caso di esito positivo, l'algoritmo di verifica della sicurezza.

Durante l'esecuzione, qualsiasi informazione di rilievo verrà stampata nello Status, in modo che l'utente possa seguire passo passo lo svolgimento delle operazioni.



Il quarto ed ultimo step visualizza un resoconto del sistema dopo l'esecuzione dei due algoritmi e offre alcune funzioni importanti, come il salvataggio del sistema sui file di testo (utile se la richiesta effettuata lo ha portato in un nuovo stato sicuro, che è diventato il nuovo sistema) o come la possibilità di effettuare una seconda richiesta sul sistema.

Si può, infine, decidere di chiudere il wizard e ritornare alla finestra principale.



Tutte le informazioni visualizzate nello status della finestra principale (log) possono essere rimosse con il pulsante "Pulisci Log" oppure possono essere salvate sul file "Log.txt" presente nella cartella del software.

Ad ogni salvataggio corrisponde un nuovo file di log che sovrascrive il vecchio.

Sempre relativa allo status è la funzione "Autoscroll", che permette un (veloce) scorrimento automatico delle informazioni man mano che vengono visualizzate nello status. A seconda del gusto dell'utente, l'autoscroll può essere attivato o disattivato in qualsiasi momento.

Nella "plancia di comando" del software vi sono altri pulsanti degni di nota, oltre ad "Esegui". I tre pulsanti "Esempio.." assegnano al sistema i dati degli esempi sul libro di testo e calcolano i due algoritmi su questi ultimi, mostrando il raggiungimento di uno stato sicuro e di due stati non sicuri.

Il tasto "Rnd!", invece, crea dei dati semi-randomici e li assegna al sistema; su questo sistema random verranno eseguiti gli algoritmi.

Anche nel caso di questi quattro pulsanti, ogni operazione compiuta sarà visibile nello status.

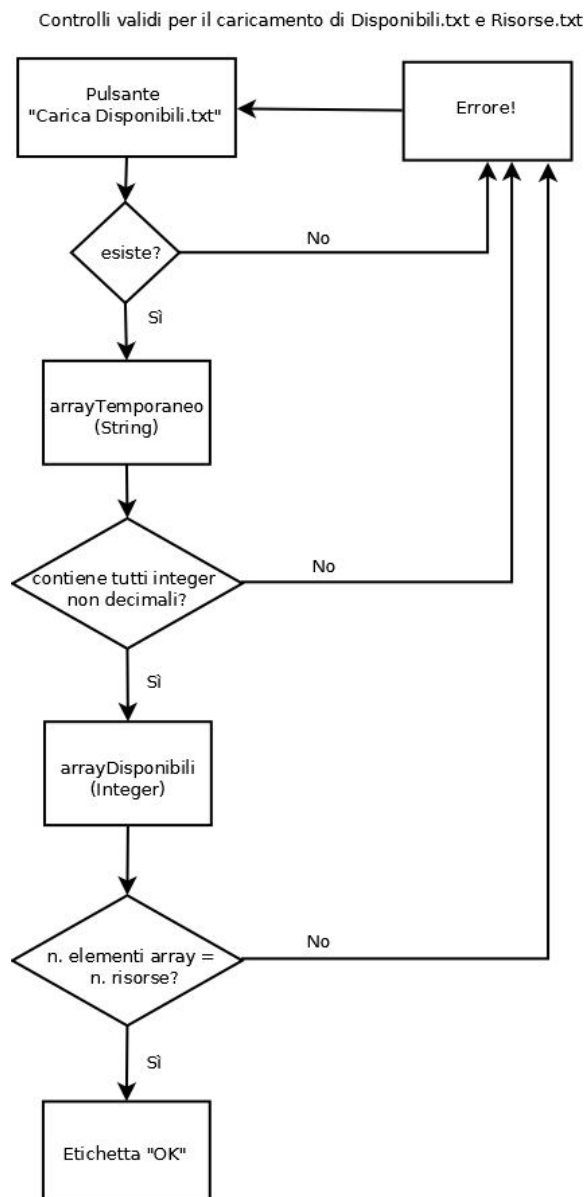
I controlli

Pur essendo un progetto che come fine ha la dimostrazione del funzionamento di un algoritmo e non la programmazione fine a sé stessa, sono stati integrati alcuni controlli basilari sull'immissione dei dati, in modo da evitare errori di digitazione o distrazione comuni che potrebbero inficiare il funzionamento degli algoritmi.

Tutte le notifiche di errore vengono stampate nello status; in più, nel wizard non è possibile procedere allo step successivo se i dati non sono corretti.

Per tutta la durata del wizard, inoltre, vi è un semplice controllo che disabilita il tasto "Esegui", rendendolo non cliccabile.

Uno dei controlli più macchinosi riguarda il caricamento dei quattro file di testo in altrettanti array mono-dimensionali di interi. A titolo esemplificativo, si guardi il diagramma di flusso del suddetto controllo. Il diagramma si riferisce a Disponibili.txt e a Risorse.txt, ma vale anche per Assegnate.txt e Massimo.txt, basta cambiare l'espressione: "n. elementi array = n. risorse?" in "n. elementi array = n. risorse * n. processi?"



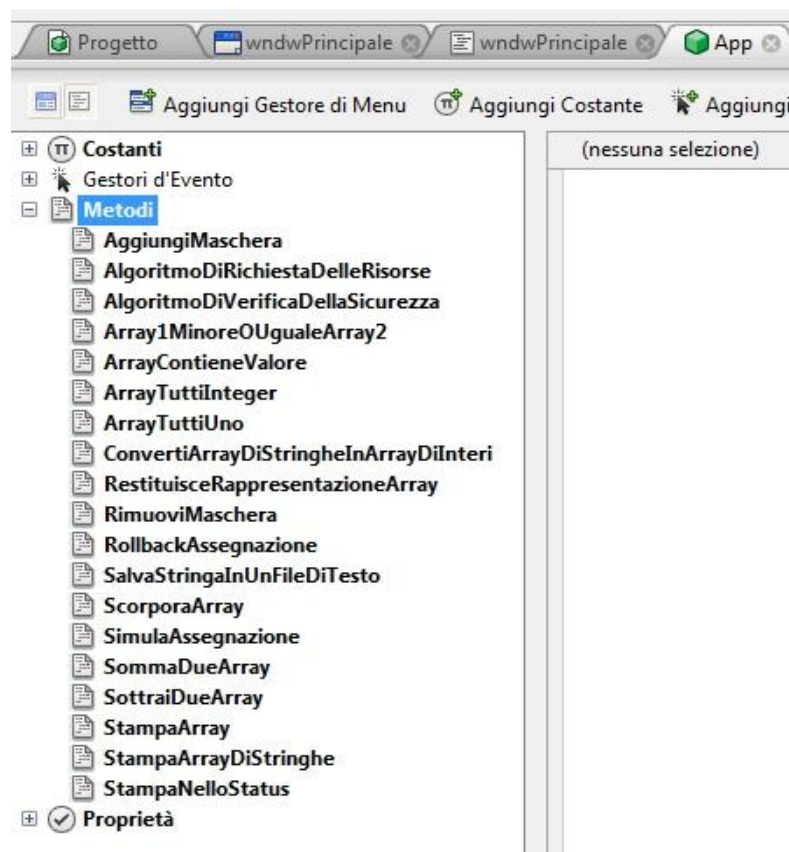
La logica del software, metodi, variabili globali

Il programma, compatibilmente con il tempo a disposizione e con la conoscenza dello studente dell'ambiente di sviluppo, è stato reso più modulare possibile, in modo da limitare la duplicazione dello stesso codice e facilitare la circoscrizione dei bugs in fase di testing. Tutto il software si basa su metodi e variabili globali, a cui si può avere accesso da qualsiasi finestra o pulsante, usando la sintassi:

App.nomeMetodo o App.nomeVariabile

I dati relativi al sistema vengono tutti quanti scritti in variabili globali, che vengono "passate" ai metodi sia direttamente che indirettamente. Alcuni metodi, infatti, necessitano che gli vengano forniti i dati necessari tramite parametri, altri invece accedono direttamente alle variabili globali.

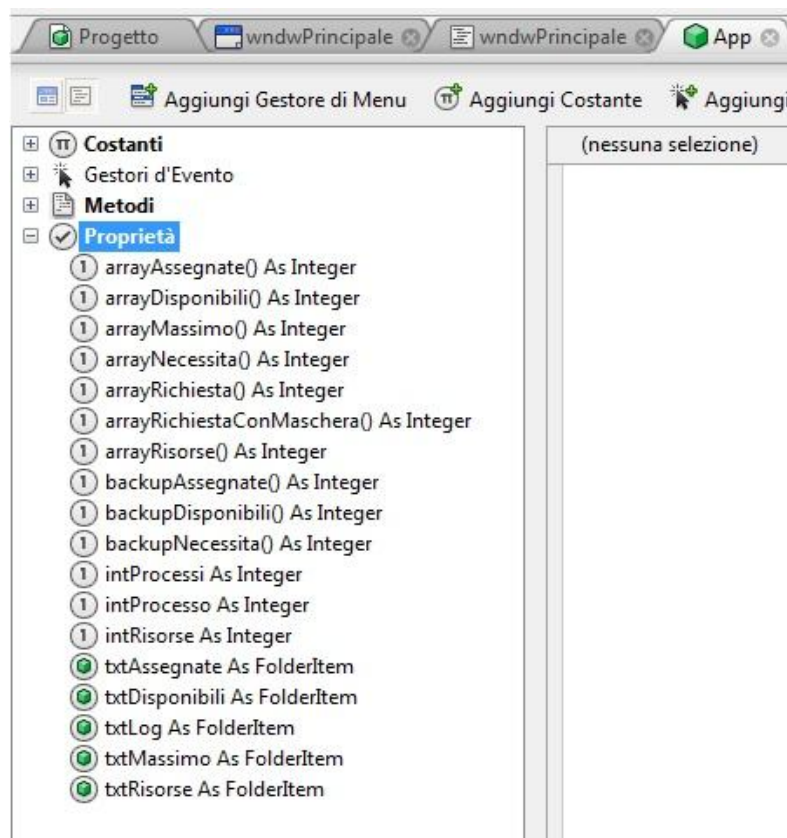
Nello screenshot seguente sono elencati tutti i metodi implementati nel software:



Essendo ogni metodo adeguatamente commentato nel codice sorgente, eviteremo di dilungarci in merito. Piuttosto, mi preme segnalare che i metodi relativi ai due algoritmi non richiedono parametri, ma sono tra i pochi ad accedere direttamente alle variabili globali. Essi vengono invocati dopo una serie di controlli e restituiscono un "True" se tutto va bene, il che li rende facilmente implementabili all'interno di cicli condizionali.

In più, la separazione di questi algoritmi in subroutine separate, rende il codice delle finestre e dei pulsanti nettamente più leggibile.

Nello screenshot seguente, invece, sono elencate le variabili globali:



La struttura dati dominante è l'array mono-dimensionale di interi.

In linea di massima, la logica del software segue il seguente schema:

- acquisizione dei dati sul sistema e salvataggio nelle variabili globali (i dati possono essere forniti dall'utente nel wizard o dal software negli esempi e nel randomizzatore);
- calcolo dell'algoritmo di richiesta delle risorse;
- calcolo dell'algoritmo di verifica della sicurezza.

Alcuni esempi

Il programma è stato testato con successo sugli esempi del libro (inclusi nel software) e su alcuni altri sistemi, tra cui:

Processi 5, Risorse 3

Risorse: 1,10,4

Richiesta: 2,0,0 per il processo 1

Assegnate: 3,3,0,2,2,0,1,2,3,0,0,1,1,1,1

Massimo: 7,5,4,6,9,3,9,4,5,7,4,5,4,3,3

Disponibili: 8,7,5