AlgoBuild 0.80 - Manuale Essenziale

Sommario

Cos'è AlgoBuild

AlgoBuild è un software didattico per progettare programmi.
Presenta un ambiente visuale in cui realizzare i flow-chart rispettando le regole della programmazione strutturata.
Il programma prodotto viene visualizzato ed è modificabile anche in forma di pseudo-codice.

Se le istruzioni vengono inserite rispettando una opportuna sintassi è possibile eseguire il programma in modalità passo-passo o temporizzata seguendo l'esecuzione delle istruzioni, il calcolo dei valori, le operazioni di input e output.

Vai al Sommario

Novità rispetto alla versione 0.75

La versione corrente, 0.80 ha apportato notevoli aggiunte rispetto alle precedenti.

Differenze rispetto alla versione precedente:
- Tipi di dati diversificati: decimali in virgola mobile (double), interi (int), stringhe (string) e logici (boolean).

- Funzioni per la manipolazione e la conversione tra dati di tipo diverso.
- Pseudo codice editabile - Anche l'area pseudo codice è utilizzabile per inserire e modificare il programma.
- Procedure e Funzioni - Oltre al programma principale (main) è possibile definire procedure e funzioni richiamabili da main o da altre procedure / funzioni. La ricorsione è supportata.
- Commenti - Un nuovo elemento, non istruzione, è rappresentato dai commenti inseribili nel diagramma.
- Annulla (Undo) - È stata aggiunta la possibilità di annullare i comandi e rifare i comandi annullati.
- Colori - Si possono modificare i colori usati nel diagramma e nella pseudo codifica: colore di sfondo, il colore di riempimento (delle forme), il colore delle linee e dei caratteri, lo spessore delle linee. Si possono impostare colori di default.
- Nuovo formato file - I programmi AlgoBuild vengono memorizzati nel nuovo formato con estensone .algobuild. Questa forma di memorizzazione, oltre alle nuove funzionalità aggiunte, permette di mantenere le informazioni di copyright dell'utente.
- Uso privato o pubblico - AlgoBuild può essere usato in modalità privata o pubblica. Privata significa che i file si possono leggere/scrivere solo dalla postazione in cui sono stati prodotti. Pubblica significa che i file possono essere letti anche da altre postazioni.
- Utente e Copyright - Un servizio di "registrazione Utente" permette di iscriversi all'elenco degli Utenti Registrati.
Gli Utenti Registrati possono memorizzare dei file "pubblici" leggibili da tutti gli utenti di AlgoBuild.
- Lingua - La lingua dell'interfaccia si può cambiare in qualsiasi momento. Attualmente disponibili Italiano e Inglese.

Novità rispetto alla versione 0.73

Differenze rispetto alla versione precedente:
- Aggiunta la gestione degli array (vettori) monodimensionali
- Scrolling automatico della finestra del flow chart durante l'esecuzione
- Spostamento della selezione dei caratteri/spessore linea in una finestrella apposita accessibile con il pulsante "a" della barra strumenti
- Miglioramento dell'allineamento delle scritte negli elementi grafici
- BUGFIX: Aggiunta l'estensione di default ".bfc" nel salvataggio/apertura file

Novità rispetto alla versione 0.72

Differenze rispetto alla versione precedente:
- Selezione Tipo Font
- Scelta dimensione font per codice e per output
- Seleziona spessore linea del grafico

Novità rispetto alla versione 0.71

Vaersione attuale: AlgoBuild 0.72.
Ultimo aggiornamento 27/04/2013

1 - Numeri pseudo casuali
È stata aggiunta la funzione random per la generazione di numeri pseudo casuali. La nuova funzione non ha argomenti e restituisce un numero maggiore o uguale a 0.0 e minore di 1.0.

Esempi:
`num=random()` assegna un numero pseudo casuale tra 0.0 e 0.99999999999999 alla variabile num.
`dado=floor(random()*6+1)` assegna uno dei sei valori a caso tra 1.0, 2.0, 3.0, 4.0, 5.0 e 6.0 alla variabile dado.

2 - Operazioni di input/output e trace
Nelle operazioni di I/O da ora viene omessa la descrizione "INPUT" e "OUTPUT" se il Trace è disabilitato.

Esempio con trace abilitato:

```
OUTPUT i: 0.0
OUTPUT i: 1.0
OUTPUT i: 2.0
ecc...
```

Esempio con trace disabilitato:

```
0.0
1.0
2.0
ecc...
```

3 - Help online / Facebook / YouTube
Sono state aggiunte altre voci al menu Aiuto:
- Collegamento all'aiuto online sul sito algobuild.com
- Collegamento alla pagina FaceBook di Algobuild
- Collegamento al canale YouTube di Algobuild

4 - BUGFIX - operatore "NOT"
Nella versione 0.71 era presente un errore di programmazione e l'operatore logico not (rappresentato con il punto esclamativo prefisso !) non veniva valutato correttamente. In questa versione l'errore è stato eliminato e il NOT logico dà i risultati aspettati.

Novità rispetto alla versione 0.7

1 - Copia immagine
Con questo comando l'immagine del diagramma viene copiata negli appunti e può essere incollata all'interno di programmi di grafica (Paint, Gimp, Photoshop ecc...), all'interno di documenti tipo office o nel notepad della LIM.

2 - BUGFIX - operatore "or"
Nella versione 0.7 era presente un errore di programmazione e l'operatore logico or (rappresentato con due sbarrette verticali ||) non veniva valutato correttamente. In questa versione l'errore è stato eliminato e l'OR logico dà i risultati aspettati.

3 - Informazioni su...
Dalla finestra di messaggio "Informazioni su..." è possibile accedere al sito algobuild.com premendo sul bottone corrispondente.

Novità rispetto alla versione 0.6

1 - Splash screen
Adesso all'avviamento il programma si presenta.
Lo splash screen è una finestra o immagine visualizzata all'avvio di un programma.
Quello di AlgoBuild riporta un messaggio di saluto (*), nome e numero di versione; è simile a quello del sito internet ufficiale di AlgoBuild.
Vai al sito algobuild.com

(*) "Hello World!" è tradizionalmente il messaggio visualizzato dal primo programma quando si impara un nuovo linguaggio.

2 - Esecuzione temporizzata
Nella barra degli strumenti sono stati introdotti tre elementi nuovi: una casella di spunta (checkbox), una casella numerica e il pulsante Pausa Esecuzione.
La casella di spunta, normalmente selezionata, riporta la dicitura "Passo passo". Quando è selezionata l'esecuzione avviene un passo alla volta come sempre. Se viene deselezionata, premendo il pulsante "Esegui istruzioni" con la freccia verde, le istruzioni verranno eseguite una di seguito all'altra con la temporizzazione stabilita dalla casella numerica "Tempo (100-5000 ms)". Come indicato il tempo tra una istruzione e la successiva va da un minimo di 100 ms (pari a 0,1 s) a un massimo di 5000 ms (pari a 5 s). I tempi sono puramente indicativi.
La pressione del pulsante "Pausa Esecuzione" fa arrestare il programma. Premendo "Esegui istruzioni" l'esecuzione riprende da dove è stata interrotta.
La modifica del tempo è recepita solo se il programma è in pausa o interrotto.

3 - Identificatori "case-sensitive"
Da questa versione c'è differenza tra maiuscolo e minuscolo, quindi x e X sono due variabili diverse.

4 - Risoluzione di un errore relativo all'interpretazione delle espressioni
Se una qualsiasi variabile aveva all'interno una parola chiave avveniva un errore.
È capitato con "distanza". Il programma si bloccava inspiegabilmente. Ciò era dovuto alla presenza del nome di funzione "tan" all'interno della variabile "distanza".

5 - Eseguibile ".exe" per Windows
Per semplificare l'utilizzo del programma, questo adesso viene distribuito anche in formato eseguibile per windows.
NOTA: il runtime Java JRE 6 (o successivo) è ancora necessario.

Vai al Sommario

Formato eseguibile e requisiti di sistema

Il programma è distribuito in due versioni: come archivio eseguibile Java e come programma eseguibile Windows.

Per poter essere utilizzato è necessario aver installato nel computer un ambiente di esecuzione Java (JRE) versione 6.
Siccome il programma è basato su oggetti e librerie Java standard è eseguibile su qualsiasi piattaforma che supporti Java 6.

La versione in formato jar è utilizzabile su Windows, Linux, Mac e qualsiasi altra piattaforma mentre la versione in formato exe è utilizzabile solo in Windows.

Vai al Sommario

Installare il programma

1 - Scompattare l'archivio zip (cartella compressa)

2 - Copiare il file AlgoBuild080.jar o AlgoBuild080.exe in una cartella.
Per comodità è opportuno che sia la cartella di lavoro.

Vai al Sommario

Avviare il programma

Il file viene distribuito con il nome AlgoBuild080.jar o AlgoBuild080.exe.

La versione eseguibile Windows parte con un doppio click del mouse.

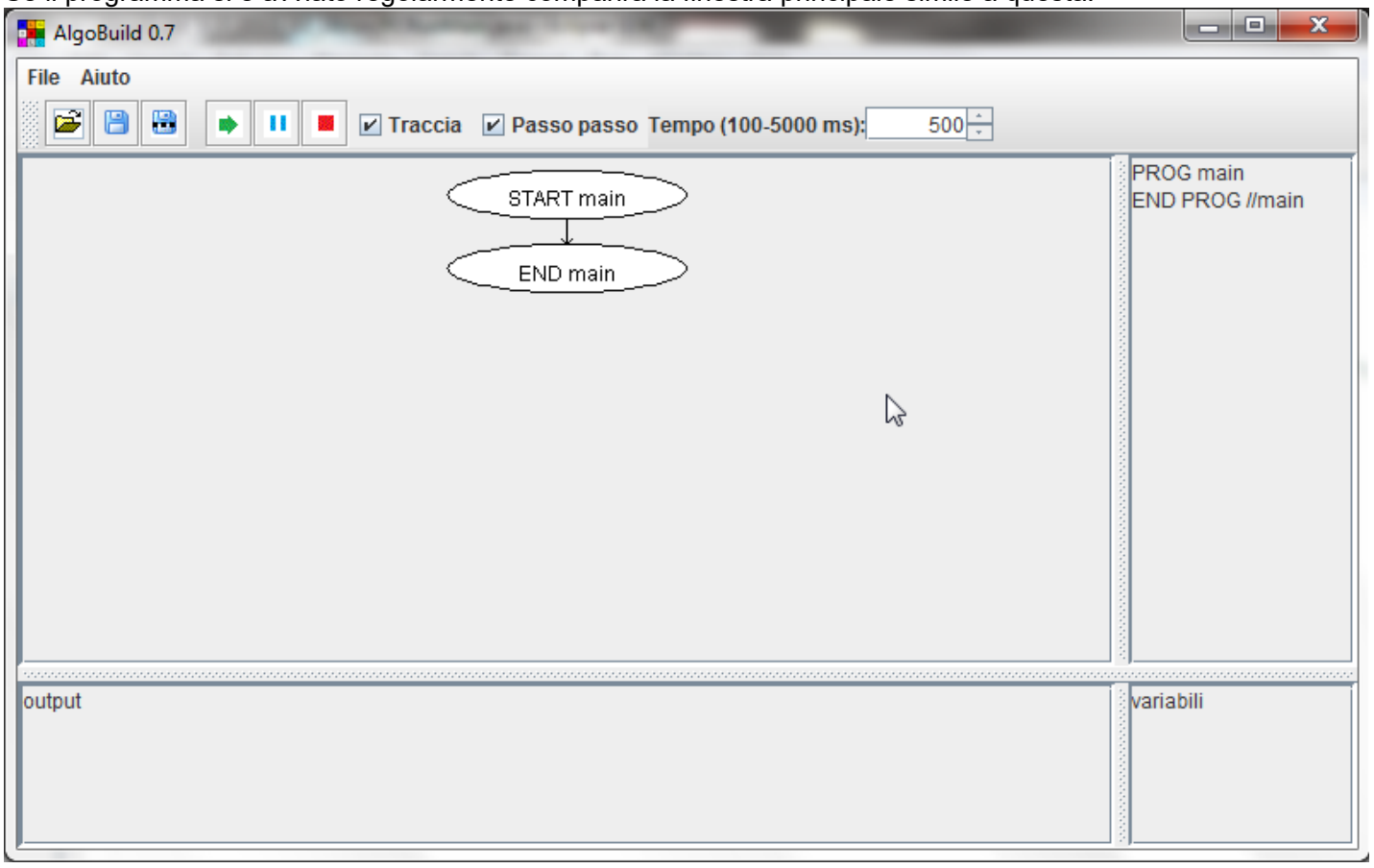
Per la versione jar il discorso è più articolato e si possono presentare tre possibilità

1. **Doppio clic:**
   È il caso (abbastanza comune) più semplice; nel sistema i file jar sono associati al runtime Java. Se questo modo non funziona passare al modo 2.

2. **Avvio da console:**
   È il caso (meno comune) sfortunato a livello 1; nel sistema i file jar NON sono associati al runtime Java. Per avviare il programma bisogna aprire un terminale (in Windows: Esegui -> cmd), spostarsi nella cartella contenente il programma e invocare java direttamente con il comando: `java -jar AlgoBuild080.jar`
   In questo caso è più opportuno aggiungere un collegamento o un file batch (shell script per i sistemi Unix/Linux) per semplificare l'operazione di avvio. Se anche questo modo non funziona passare al modo 3.

3. **Avvio da console con path completo:**
   È il caso (ancora meno comune) sfortunato a livello 2; nel sistema i file jar NON sono associati al runtime Java e il programma java NON è nel PATH di ricerca del sistema operativo.
   Per avviare il programma bisogna aprire un terminale (in Windows: Esegui -> cmd), spostarsi nella cartella contenente il programma e invocare java direttamente specificando tutto il path:
   `C:\Programmi\Java\jre6\bin\java -jar AlgoBuild080.jar`
   Naturalmente il percorso C:\Programmi\Java\jre6\bin\ dipende da dove è stato installato jre nel sistema in questione.
   Anche in questo caso è più opportuno aggiungere un collegamento o un file batch (shell script per i sistemi Unix/Linux) per semplificare l'operazione di avvio.

Se il programma si è avviato regolarmente comparirà la finestra principale simile a questa:

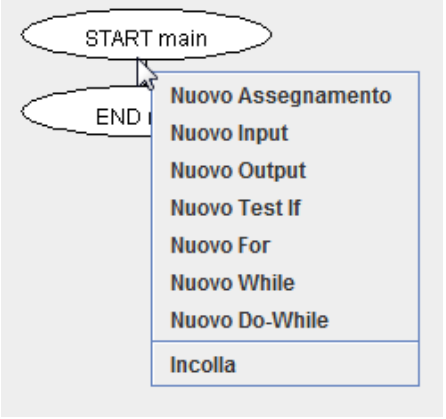


Vai al Sommario

Inserire una istruzione

Tra le ellissi Start e End si trova una freccia verso il basso.
Spostandovi il mouse sopra la freccia viene sostituita da un quadrato rosso.

Premendo il pulsante sinistro del mouse compare un menu da cui è possibile scegliere l'istruzione da inserire.



Nota: ogni freccia verso il basso è un punto in cui è possibile inserire una istruzione.

Vai al Sommario

Modificare una istruzione

Spostare il puntatore del mouse sopra l'istruzione da modificare, premere il pulsante sinistro del mouse e scegliere Modifica dal menu.

Vai al Sommario

Copiare/tagliare una istruzione

Spostare il puntatore del mouse sopra l'istruzione da modificare, premere il pulsante sinistro del mouse e scegliere Copia o Taglia dal menu.

Effetti:

- Copia invia negli appunti locali una copia dell'istruzione scelta.
- Taglia invia negli appunti locali una copia dell'istruzione scelta ed elimina l'istruzione stessa dal punto in cui si trovava.

Vai al Sommario

Incollare una istruzione dagli appunti

Spostare il puntatore del mouse sopra a un punto di inserimento (freccia verso il basso) premere il pulsante sinistro del mouse e scegliere Incolla dal menu.

Vai al Sommario

Salvare il lavoro

Dal menu principale scegliere File -> Salva... oppure File -> Salva con nome...

In alternativa è possibile schiacciare uno dei pulsanti con il floppy disk sulla barra degli strumenti.

Vai al Sommario

Aprire un lavoro dal disco

Dal menu principale scegliere File -> Apri...

In alternativa è possibile schiacciare il pulsante con la cartellina sulla barra degli strumenti.

Vai al Sommario

Esportare un diagramma in formato immagine

Dal menu principale scegliere File -> Salva immagine

Vai al Sommario

Selezionare font, dimensione caratteri, spessore della linea

Sulla barra degli strumenti schiacciare il bottone con l'icona "a".

Comparirà una finestrella da cui è possibile scegliere:
- Nome/tipo del carattere da usare
- Dimensioni caratteri della vista "codice"
- Dimensioni caratteri della finestra output e della zona delle variabili
- Colore di sfondo della zona flowchart e della zona pseudo codice.
- Colore di riempimento della zona flowchart e colore istruzioni della zona pseudo codice.
- Colore della linea.
- Spessore della linea usata per il disegno. Due pulsanti permettono di salvare le preferenze o ripristinare i valori predefiniti.

Vai al Sommario

Eseguire un programma passo-passo

NOTA: per eseguire un programma in modalità passo passo, cioè una istruzione alla volta, è necessario che la casella "Passo passo" sia selezionata.

Premere il pulsante con la freccia verde e i puntini sulla barra degli strumenti.

Comparirà una finestrella da cui è possibile scegliere:
- Traccia sì / no,
- Passo passo - sì / no,
- Temporizzazione (ms).

Ad ogni pressione del pulsante con la freccia verde l'esecuzione del programma avanza di una istruzione. Alcune istruzioni (test e cicli) necessitano anche di più pressioni in quanto deve essere anche valutata la condizione o incrementate le variabili (for).

La casella di spunta "Traccia" riporta in output ogni messaggio e lo stato delle variabili a ogni istruzione. Togliendo la spunta vengono visualizzati solo input e output.

Per interrompere l'esecuzione del programma prima di arrivare a END premere il pulsante con il quadrato rosso.
NOTA: Se non si interrompe l'esecuzione non è possibile modificare il programma.

NOTE

Se durante l'esecuzione vengono trovati degli errori (es. espressioni errate) il programma termina riportando un messaggio di errore.

Vai al Sommario

Eseguire un programma in modalità temporizzata

NOTA: per eseguire un programma in modalità temporizzata, cioè una istruzione dietro l'altra a intervalli di tempo regolari, è necessario che la casella "Passo passo" sia deselezionata.

Premere il pulsante con la freccia verde sulla barra degli strumenti.

Il programma si avvia e avanza di una istrusione alla volta a intervalli regolari.
Alcune istruzioni (test e cicli) necessitano anche di più tempo in quanto deve essere anche valutata la condizione o incrementate le variabili (for).

Se durante l'esecuzione si preme il pulsante "Pausa Esecuzione" il programma viene fermato momentaneamente. Premendo "Esegui Istruzioni" riprende da dove era stato interrotto.
La modifica del tempo è recepita solo se il programma è in pausa o interrotto.

La casella di spunta "Traccia" riporta in output ogni messaggio e lo stato delle variabili a ogni istruzione. Togliendo la spunta vengono visualizzati solo input e output.

Per interrompere l'esecuzione del programma prima di arrivare a END premere il pulsante con il quadrato rosso.

Se non si interrompe l'esecuzione non è possibile modificare il programma.

NOTE

Se durante l'esecuzione vengono trovati degli errori (es. espressioni errate) il programma termina riportando un messaggio di errore.

Vai al Sommario

Sintassi

Si riportano in breve le principali regole e la sintassi delle istruzioni.

- **Tipi di dati**: ogni espressione e ogni variabile ha associato un *tipo*.
  I tipi sono: int (numeri interi), double (numeri decimali a doppia precisione), string (sequenze di caratteri), boolean (logico true/false). Il modo di scrivere una espressione determina anche il suo tipo.
  - **int**: sono numeri scritti come sequenza di cifre senza il punto oppure espressioni algebriche contenenti SOLO int e variabili di tipo int.
    Esempi: 3, 10*2+4, 3*k (se k è variabile int).
    numero = 71 (numero viene definita come variabile di tipo intero).
    Numeri interi in basi diverse
    numero = 10 (valore in base 10 ... cioè dieci!)
    numero = 010 (valore ottale - base 8 = decimale 8)
    numero = 0x10 (valore esadecimale = decimale 16)
    numero = 0b10 (valore binario = decimale 2)
  - **double**: sono numeri scritti come sequenza di cifre con il punto oppure espressioni algebriche contenenti ALMENO un double o una variabile di tipo double.
    Esempi: 5.6, 10*2-0+4, 9*x (se x è variabile double).
    numero = 44.0 (numero viene definita come variabile di tipo double).
    Si possono scrivere anche in notazione scientifica:
    1.2e5 = 1.2 * 10^5 = 12000
    1.2e-5 = 1.2 * 10^-5 = 0.000012
  - **string**: sono sequenze di caratteri racchiuse tra doppi apici.
    Esempi: "ciao", "tanti quanti", "torno\nsubito" (il simbolo \n inserisce un "a capo").
    nome = "Mondo"
    messaggio = "Ciao " + nome + "!"
  - **boolean**: possono avere solo due valori true e false.
    condizione = true
    test = x>y
- **Nomi di variabili**: possono essere scritte in minuscolo, MAIUSCOLO o modoMisto.
  Devono iniziare per carattere alfabetico, possono contenere cifre o trattini di sottolineatura.
  NON possono contenere spazi o caratteri speciali.
  CONSIGLIO: usare nomi tutti minuscoli o "a cammello" con iniziale minuscola.
  ESEMPI: numero, prezzo, nElementi.
  Attenzione:C'È DIFFERENZA tra Maiuscolo e minuscolo.
- **Array** (vettori): Si utilizzano come i nomi di variabili seguiti da un indice (numero o espressione) tra una coppia di parentesi quadre.
  Esempio: *a[0]=7* oppure *b[i]=a*3* oppure *x=v1[n+k*2]*
  NOTA BENE: L'indice è un valore numerico che deve essere di tipo intero.
  L'indice minimo è 0.
- **Assegnamento**:
  variabile=espressione oppure
  vettore[indice]=espressione
  Esempi: *x=7* oppure *x=x+1* oppure *giorni[mese]=31*
  In un blocco di assegnamento si possono inserire più istruzioni una per riga.
  Il tipo della variabile dipende dall'espressione a destra del simbolo di assegnamento (uguale).

NOTA: Ad una variabile esistente non può essere assegnata una espressione di tipo diverso.
Unica eccezione double che accetta espressioni di tipo int.

- **Input**: legge un valore da tastiera. Se non specificato i valori sono di tipo double.
  Esempi:
  lunghezza (legge un numero double e lo inserisce nella variabile lunghezza)
  valori[0] (legge un numero double e lo inserisce alla prima posizione del vettore valori)
  Per leggere un input di un tipo specifico si deve premetterne il nome.
  int nElementi (legge un numero intero e lo inserisce nella variabile nElementi)
  string nome (legge una sequenza di caratteri e la inserisce nella variabile nome)
  In un blocco di input si possono inserire più istruzioni di lettura una per riga.

- **Output**: visualizza il valore di una variabile o di una espressione nel riquadro Output (console).
  Esempi:
  nElementi
  *nElementi*4*
  "ciao " + nome
  In un blocco di output si possono inserire più istruzioni di visualizzazione una per riga.

- **If**: valuta una espressione booleana e prosegue sul ramo true (T) se la condizione è vera, prosegue sul ramo false (F) altrimenti.
  Esempi:
  y < 5
  *x == z*2*
  a[i] < a[i+1]

- **While**: ciclo precondizionale. Valuta una espressione booleana ed esegue un gruppo di istruzioni finché la condizione è vera.
  Espressione booleana come nella istruzione if.

- **Do-While**: ciclo postcondizionale. Esegue un gruppo di istruzioni e alla fine valuta una espressione booleana. Ripete finché la condizione è vera.
  Espressione booleana come nella istruzione if.

- **For**: è un ciclo usato come contatore. Si usa una variabile in tre parti della stessa istruzione: inizializzazione, condizione, aggiornamento.
  La parte "inizializzazione" è un assegnamento come: *i = 0*.
  La parte "condizione" è una espressione booleana come: *i <= 10*.
  La parte "aggiornamento" è un assegnamento per calcolare il valore successivo come: *i = i+1*.

- **Espressioni**:
  - **Operatori aritmetici**
    - + addizione definita per i tipi int, double e string
    - - sottrazione definita per i tipi int e double
    - * moltiplicazione definita per i tipi int e double
    - / divisione definita per i tipi int e double
    - % modulo (resto della divisione tra interi) definita solo per i tipi int. Esempio 12 % 7 restituisce 5
    - ^ potenza (es. x^2) definita solo per i tipi double
    - & (And aritmetico bit a bit - solo tra interi) definita solo per i tipi int. Esempio 7 & 12 restituisce 4
    - | (Or aritmetico bit a bit - solo tra interi) definita solo per i tipi int. Esempio 7 | 12 restituisce 15
    - >> (shift bit a destra) definita solo per i tipi int. Esempio: 16 >> 2 restituisce 4
    - << (shift bit a sinistra) definita solo per i tipi int. Esempio: 16 << 2 restituisce 64
  - **Operatori logici**:

- == (uguaglianza) definita per i tipi int, double e string restituisce boolean
- != (disuguaglianza) definita per i tipi int, double e string restituisce boolean
- > (maggiore di) definita per i tipi int, double e string restituisce boolean
- < (minore di) definita per i tipi int, double e string restituisce boolean
- >= (maggiore o uguale a) definita per i tipi int, double e string restituisce boolean
- <= (minore o uguale a) definita per i tipi int, double e string restituisce boolean
- && (And logico) definita per i tipi boolean restituisce boolean
- || (Or logico) definita per i tipi boolean restituisce boolean
- ! (negazione logica) definita per i tipi boolean restituisce boolean

◦ **Funzioni matematiche**
I valori di x di tipo double, valori di ritorno di tipo double.

- sqrt(x) radice quadrata di x
- sin(x) funzione trigonometrica seno di x (x in radianti)
- cos(x) funzione trigonometrica coseno di x (x in radianti)
- tan(x) funzione trigonometrica tangente di x (x in radianti)
- cotan(x) funzione trigonometrica cotangente di x (x in radianti)
- asin(x) funzione trigonometrica inversa arcoseno di x
- acos(x) funzione trigonometrica inversa arcocoseno di x
- atan(x) funzione trigonometrica inversa arcotangente di x
- acotan(x) funzione trigonometrica inversa arcocotangente di x
- sinh(x) funzione iperbolica
- cosh(x) idem
- tanh(x) idem
- exp(x) e (numero di eulero) elevato alla ...
- ln(x) logaritmo naturale in base e
- log10(x) logaritmo in base 10 di x
- abs(x) valore assoluto x
- ceil(x) arrotondamento intero per eccesso
- floor(x) arrotondamento intero per difetto
- round(x) arrotondamento all'intero più vicino
- random() numero casuale tra 0.0 e 0.99999999999999

Funzioni di manipolazione stringhe e conversione tra tipi diversi

- strlen(s) - Lunghezza della stringa s. Ritorna il numero (intero) di caratteri di s. Esempio: lung = strlen("ciao") restituisce 4.
- strchar(s,p) - Carattere dalla stringa s. Ritorna una stringa formata dal carattere di s alla posizione p. La prima posizione è 0. Esempio: ch = strchar("ciao",2) - restituisce "a".

- substr(s,i,j) - Sottostringa di s. Ritorna una sottostringa dalla posizione i alla j-1. La prima posizione è 0. Esempio: nome = strchar("ciao Mondo!",5,10) - restituisce "Mondo".
- strpos(s,t) posizione della sottostringa ritorna il numero (intero) con la posizione di t in s la prima posizione Ë 0 esempio: pos = strpos("ciao Mondo!","Mondo") restituisce 5
- strupr(s) converte in maiuscolo ritorna la stringa convertita in maiuscolo esempio: maiu = strupr("Ciao") restituisce "CIAO"
- strlwr(s) converte in maiuscolo ritorna la stringa convertita in maiuscolo esempio: maiu = strlwr("Ciao") restituisce "ciao"
- trim(s) elimina spazi iniziali e finali ritorna la stringa senza spazi a inizi / fine esempio: saluto = trim(" Ciao ") restituisce "Ciao"
- ctoi(s) codice intero del carattere ritorna il numero (intero) codice ASCII del primo carattere di s esempio: cod = ctoi("Ciao") restituisce 67
- itoc(i) carattere dal codice intero ritorna il carattere corrispondente al codice ASCII (intero) passato esempio: char = itoc(65) restituisce "A"
- stoi(s) converte stringa in intero ritorna il numero (intero) convertito dalla stringa data esempio: val = stoi("12") restituisce il numero intero 12
- itos(i) rappresentazione in stringa del numero ritorna la stringa corrispondente al numero (intero) passato esempio: s = itos(65) restituisce "65"
- iformats(i,n) formattazione del numero in stringa ritorna la stringa lunga l corrispondente al numero (intero) passato con eventuali spazi iniziali esempio: s = itos(65,5) restituisce " 65"
- sbasetoi(s,b) converte stringa in intero espresso in una base numerica (da 2 a 36) ritorna il numero (intero) convertito dalla stringa nella base data esempio: val = stoi("12", 16) restituisce il numero intero 18
- ibasetos(i,b) rappresentazione in stringa del numero in una base numerica (da 2 a 36) ritorna la stringa corrispondente al numero (intero) passato nella base specificata esempio: s = itos(10,16) restituisce "A"
- stod(s) converte stringa in double ritorna il numero (double) convertito dalla stringa data esempio: val = stod("12.45") restituisce il numero double 12.45
- dtos(d) rappresentazione in stringa del numero ritorna la stringa corrispondente al numero (double) passato esempio: s = dtos(65.33) restituisce "65.33"
- dformats(d,n,m) formattazione del numero double in stringa ritorna la stringa lunga n con m decimali corrispondente al numero (double) passato con eventuali spazi iniziali esempio: s = itos(65.1234,8,2) restituisce " 65.12"
- dtoi(d) conversione da double a intero ritorna il numero intero eliminando i decimali dal numero (double) passato esempio: i = dtoi(65.33) restituisce il numero intero 65
- type(v) informazioni sul tipo di dato ritorna una stringa con il nome del dato corrispondente esempi type("ciao") restituisce "(string)" type(123.456) restituisce "(double)" type(123) restituisce "(int)"

- **Constanti** e valori particolari: (non possono essere usati come nomi di variabili)
  - PI pigreco
  - EULER numero di eulero
  - INFINITY più infinito
  - -INFINITY meno infinito
  - NaN not a number (0/0)

- TRUE equivale a 1.0
- FALSE equivale a 0.0

Vai al Sommario
