I diagrammi a blocchi

I diagrammi a blocchi sono degli insiemi di simboli che permettono una visualizzazione grafica degli algoritmi. Questo permette una comprensione dei processi molto veloce perché vengono immediatamente all'occhio le relazioni fra i vari componenti permettendoci di avere, rispetto al codice, una visione d'insieme dell'algoritmo molto più solida.

Ma visto che i diagrammi di flusso sono così comodi perché non li utilizziamo per programmare al posto dei linguaggi?

I vantaggi dei diagrammi sono innegabili ma quando ci ritroviamo a strutturare delle architetture software complesse l'estensione di queste mappe le rende inutilizzabili. Il codice al contrario è più difficile da comprendere ma occupa molto meno spazio ed è molto più facile da organizzare.

Seguentemente intenderò con "processo" l'attenzione sui blocchi.

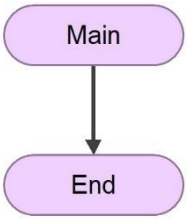
Blocchi principali

I blocchi principali sono:

Il blocco di inizio/fine

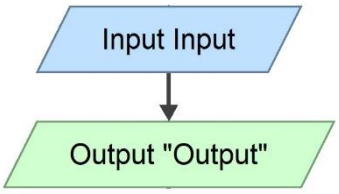
I blocchi di inizio/fine algoritmo servono ad indentificare le condizioni di partenza dell'algoritmo e dove l'algoritmo si considera concluso.

(Esistono algoritmi che non hanno fine?)



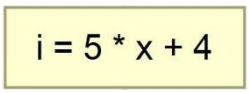
I blocchi di trasferimento delle informazioni

Servono a comunicare con ciò che è all'esterno dell'algoritmo mentre viene eseguito, rappresentano un canale di comunicazione fra il processo e l'utente che lo sta utilizzando.



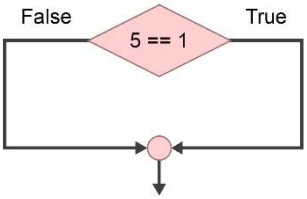
Il blocco di assegnazione o di elaborazione

Si utilizza per aggiornare i dati utilizzati dall'algoritmo, dall'assegnare ad una variabile il suo valore iniziale fino a definire il risultato di una complessa equazione.



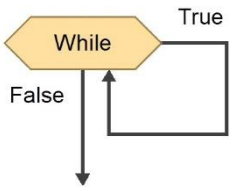
Il blocco di decisione

Il blocco di decisione permette di indirizzare il flusso del processo verso la soluzione che stiamo cercando. All'interno del blocco viene posta una domanda logica a cui il processo deve rispondere con un assenso o con un dissenso, come programmatori dobbiamo capire come gestire, o non gestire, i dati a seconda della risposta.



Il blocco iterativo

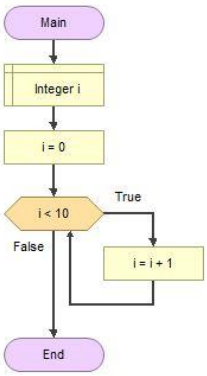
Il blocco iterativo permette di ripetere un'operazione fintanto che la condizione posta all'interno del blocco risulta vera, in caso contrario si dice che il processo esce dal ciclo.



Dopo aver dato una breve panoramica sui diagrammi a blocchi passerò a rappresentare gli algoritmi di base che vi permettano di progettare costruzioni più articolate.

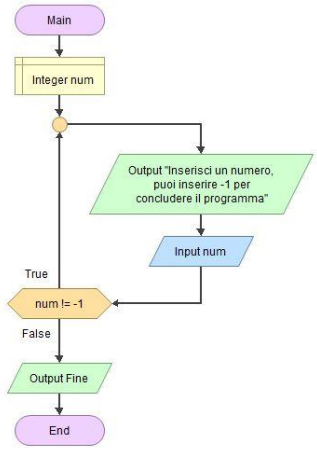
Algoritmi di base più utilizzati

Ciclo con condizione definita



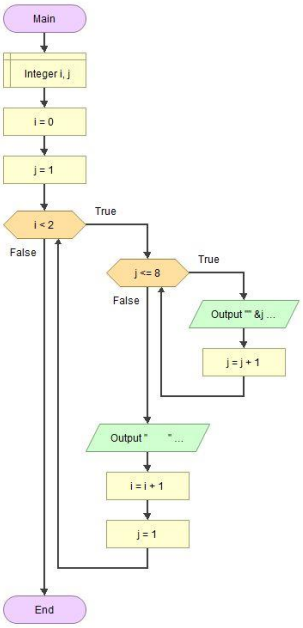
Questo è il modo più comune di utilizzare un ciclo, si costituisce un contatore che si occupa di contare il numero di iterazioni eseguite, aggiunto un determinato numero limite, in questo caso i < 10, il ciclo si conclude.

Ciclo con condizione indefinita



Il ciclo a condizione indefinita, diversamente da quello a condizione definita, si basa sull'accadimento o meno di un determinato evento. Nell'esempio in questione l'evento è l'inserimento di un "-1" da parte dell'utente.

Cicli annidati



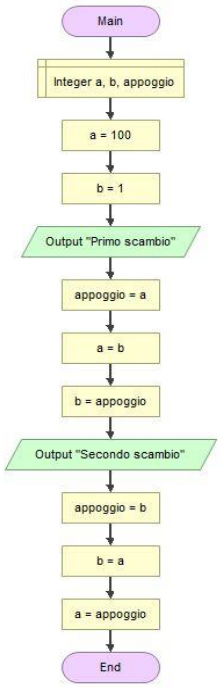
I cicli annidati servono mediamente per ripetere una sequenza più volte, magari all'avvenire di un determinato evento.

Nell'esempio che vi presento l'algoritmo mostra in output 2 volte la sequenza 12345678 in questo modo:

12345678 12345678

Il primo ciclo definisce il numero di volte che verrà mostrata la sequenza, il secondo ciclo creerà la sequenza stessa.

Scambio del valore di due variabili

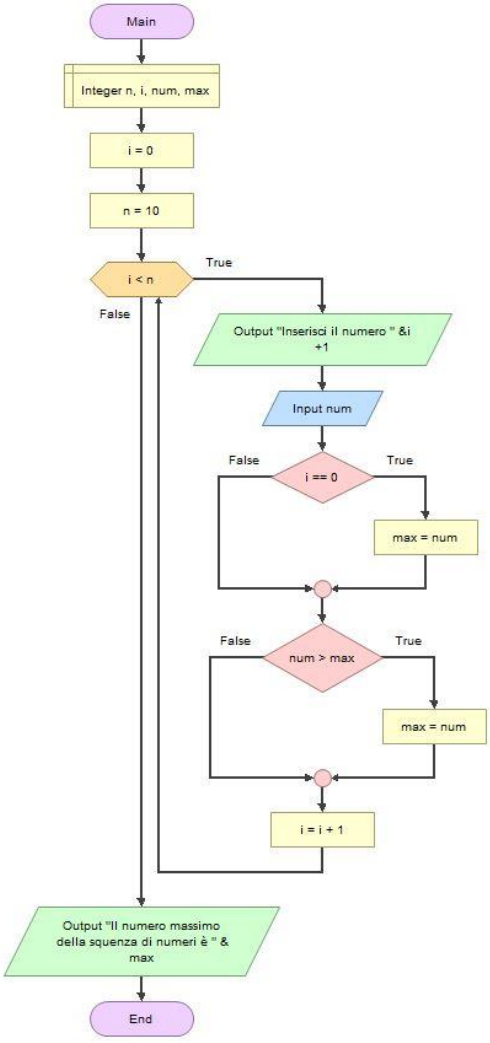


Il modo più comodo che abbiamo per scambiare due variabili a e b è inevitabilmente quello di averne una terza $appoggio$ che assuma il compito di ricordare il valore precedente di una delle altre due per non perderlo nell'operazione.

Notare che lo scambio delle variabili può avvenire in entrambi i sensi: possiamo salvare per primo a o b senza avere ripercussioni sulla logica.

Secondo voi esiste un modo per scambiare le due variabili che non utilizzi l'appoggio?

Ricerca di un numero massimo (o minimo) in una sequenza di dieci numeri inserita da utente.



Questo algoritmo cerca il numero massimo fra 10 inseriti da utente. Il **blocco di decisione** (condizione) $num > max$ permette all'algoritmo di trovare il numero maggiore e successivamente salvarlo con il **blocco elaborazione** $max = num$.

Notate che il blocco di decisione $i == 0$ serve a porre il primo numero inserito come max: se scegliessimo un numero da assegnare a max che non è stato scelto dall'utente potremmo scegliere un valore più grande dei numeri presi in input perdendo la bontà dell'algoritmo.

Per cercare il minimo basta sostituire la condizione $num > max$ con $num < min$ (dichiarando la variabile min).