Lezione 2

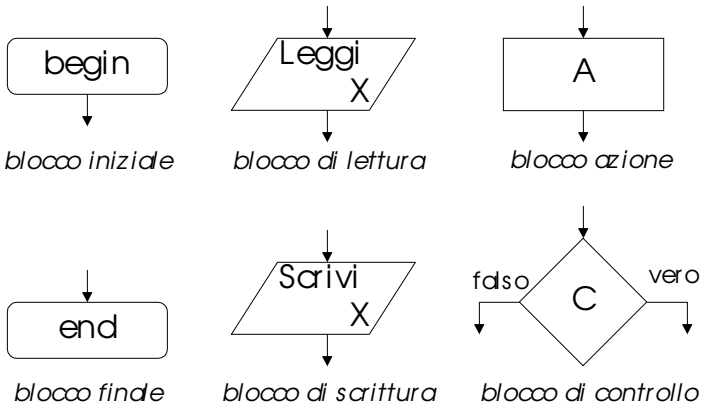
Diagrammi a blocchi
Analisi strutturata

Diagrammi a blocchi

- E' un linguaggio formale di tipo grafico per rappresentare gli algoritmi
- Attraverso il diagramma a blocchi (o *flow chart*) si può indicare l'ordine di esecuzione delle istruzioni.
- Un particolare simbolo grafico detto *blocco elementare* è associato ad ogni tipo di istruzione elementare
- I blocchi sono collegati tra loro tramite frecce che indicano il susseguirsi delle istruzioni

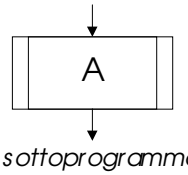
Diagramma a blocchi

- I blocchi elementari sono:



Sottoprogramma

- Con il blocco



- si indica un sottoprogramma di cui è nota la rappresentazione in diagramma a blocchi

Diagramma a blocchi

- Un diagramma a blocchi descrive un algoritmo se:
  - ha un blocco iniziale e uno finale
  - è costituito da un numero finito di blocchi azione e/o blocchi lettura/scrittura e/o blocchi di controllo
  - ciascun blocco elementare soddisfa le *condizioni di validità*

Diagramma a blocchi

- Condizioni di validità
  - ciascun blocco azione, lettura/scrittura ha una sola freccia entrante e una sola freccia uscente
  - ciascun blocco di controllo ha una sola freccia entrante e due uscenti
  - ciascuna freccia entra in un blocco o si innesta su una altra freccia
  - ciascun blocco è raggiungibile dal blocco iniziale
  - il blocco finale è raggiungibile da qualsiasi altro blocco

Esercizio

- Scrivere un algoritmo e rappresentarlo tramite un diagramma a blocchi per i seguenti problemi:
  - attraversare la strada
  - calcolare l'area del triangolo
  - trovare il max di due numeri
  - moltiplicare due numeri (usando solo l'operazione di somma)

Analisi strutturata

- Analisi volta alla stesura di descrizioni di algoritmi tramite *diagrammi a blocchi di tipo strutturato*
- Un diagramma a blocchi strutturato è più facilmente comprensibile e modificabile
- In un diagramma strutturato non apparirà mai una istruzione di salto incondizionato

Analisi strutturata

- Teorema di Bohm-Jacopini

  Ogni diagramma a blocchi non strutturato è sempre trasformabile in un diagramma a blocchi strutturato ad esso equivalente

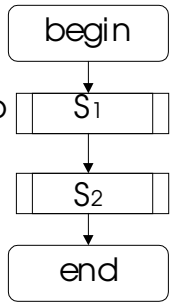
- dove due diagrammi sono equivalenti se partendo dagli stessi dati iniziali producono gli stessi risultati

Analisi strutturata

- Una descrizione è di tipo strutturato se i blocchi sono collegati tramite i seguenti schemi di flusso strutturato:
  - schema di sequenza
  - schema di selezione
  - schema di iterazione

Analisi strutturata

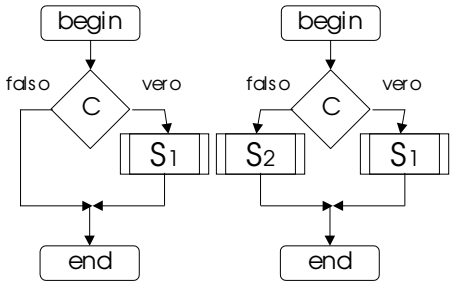
- Schema di sequenza
  - due o più schemi di flusso sono eseguiti in successione
- Nota: lo schema di sequenza è strutturato se e solo se lo sono i blocchi S_1 e S_2



Analisi strutturata

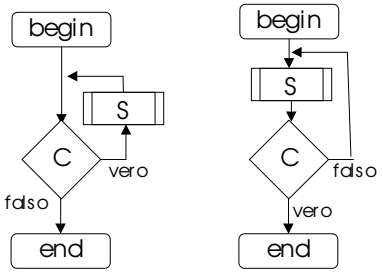
- Schema di selezione

esiste un blocco di controllo che permette di scegliere quale schema di flusso debba essere eseguito tra due schemi, in funzione del valore di verità del controllo



Analisi strutturata

- Schema di iterazione *(ciclo o loop)*
  - modo conciso per descrivere azioni che devono essere ripetute



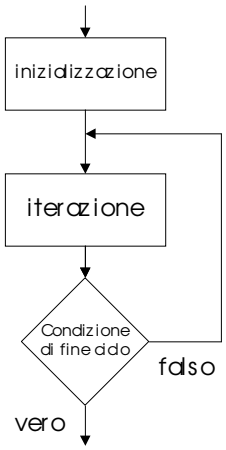
Analisi strutturata

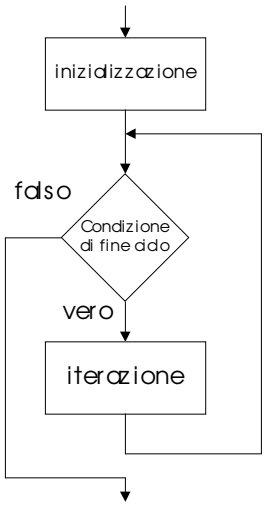
- Nota:
  - I due schemi non sono equivalenti: in un caso lo schema S è eseguito almeno una volta e nell'altro potrebbe non essere mai eseguito
  - La condizione vero/falso per il controllo possono essere invertite: si parla di *iterazione per vero* quando S è eseguito finché la condizione su C è vera e *iterazione per falso* nell'altro caso

Note sullo schema di iterazione

- Quando è necessario eseguire lo stesso insieme di operazioni più volte si adotta un particolare schema di iterazione:
  - è costituito da una sequenza di azioni di assegnazione dette *istruzioni di inizializzazione*
  - una iterazione (ripetizione) di una sequenza di azioni *(iterazione)* per un numero specificato di volte

Analisi strutturata





Analisi strutturata

- Condizione di fine ciclo: viene controllata dopo l'esecuzione di ogni blocco di iterazione
- può essere con *controllo in coda* al ciclo o con *controllo in testa*

Analisi strutturata

- Un ciclo è detto *enumerativo* quando è noto a priori il numero di volte che deve essere eseguito
  - si usa la *tecnica del contatore* per controllarne l'esecuzione: si usa cioè una variabile detta *contatore del ciclo* che viene inizializzata opportunamente prima di iniziare il ciclo e poi viene incrementata (o decrementata) fino a raggiungere un valore prefissato, permettendo così di eseguire una iterazione un numero specificato di volte

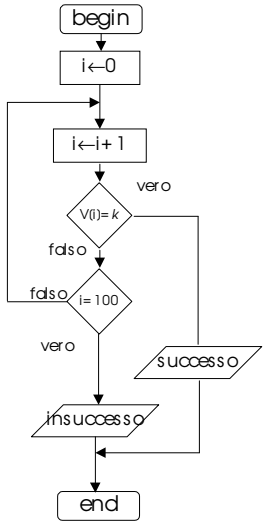
Analisi strutturata

- Un ciclo e *indefinito* quando non è noto a priori il numero di volte che deve essere eseguito
- Questo accade quando la condizione di fine ciclo dipende dal valore di una o più variabili che o dipendono dall'interazione con l'esterno o vengono modificate all'interno dell'iterazione in modo complesso

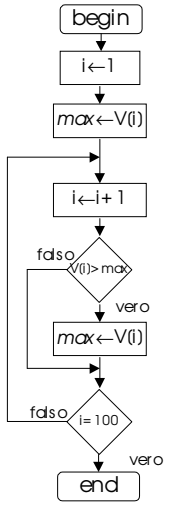
Esercizi

- Ricerca di un elemento in un vettore
- Determinare il massimo numero in un vettore
- Media di un vettore

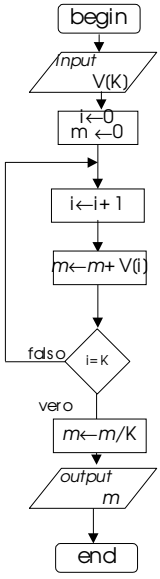
Soluzione: Trova



Soluzione: Max



Soluzione: Media



I programmi

- Per fare sì che un algoritmo sia effettivamente utilizzabile da un esecutore automatico occorre eliminare le ambiguità circa la codifica dei dati e l'interazione con gli esseri umani
- in generale durante la stesura di un programma ci si deve preoccupare dei limiti nell'intervallo di rappresentazione dei numeri, della durata non nulla delle operazioni, etc
- La soluzione integrata di queste problematiche e di quelle dell'algoritmo in sé risulta complessa e richiede alta competenza

Linguaggi di alto livello

- Per facilitare la stesura dei programmi sono stati definiti linguaggi di programmazione di alto livello che
  - permettono di descrivere le soluzioni dei problemi ad un livello di astrazione di poco inferiore a quanto visto fino ad ora per gli algoritmi
  - permettono di descrivere le operazioni di ingresso/uscita
  - permettono di definire il tipo dei dati
  - sono traducibili automaticamente in linguaggio macchina

Diversità dei linguaggi

- Sono stati sviluppati diversi linguaggi
  - Fortran, Lisp, Cobol, Basic, Pascal, C, C++, Java, Prolog
- i linguaggi si caratterizzano per
  - sintassi: l'insieme delle regole che specificano come comporre istruzioni ben formate
  - semantica: l'insieme delle regole che specificano come associare ad una istruzione una azione da compiere
- la diversità fra i vari linguaggi può consistere nella sintassi (le stesse azioni vengono descritte con termini diversi) o nella semantica

VBA

Il Visual Basic for Application

- La documentazione
- Le costanti e le variabili
- L'assegnazione
- Gli operatori

L'arte della programmazione

- La soluzione di un problema tramite un programma è un procedimento che non si esaurisce solo nello scrivere linee di codice in un dato linguaggio di programmazione, ma comprende una fase di progetto che precede e di verifica che segue la scrittura del codice

L'arte della programmazione

- Definizione del problema
- Algoritmo per la soluzione del problema
- Codifica
- Debugging
- Validazione
- Documentazione
- Manutenzione

Definizione del problema

- Definizione degli ingressi e delle uscite
  - quali variabili
  - quale dominio per ogni variabile
- Risoluzione delle ambiguità
- Scomposizione in problemi più semplici

Algoritmo

- Soluzione in pseudocodice
- Soluzione in diagramma a blocchi strutturato

Codifica

- Traduzione dell'algoritmo in istruzioni del linguaggio di programmazione

Debugging

- Correzione degli *errori sintattici* e *semantici*
- Errori sintattici
  - espressioni non valide o non ben formate nel linguaggio di programmazione
- Errori semantici
  - comportamento non aderente alle aspettative/alla intenzionalità del programmatore

Validazione

- Test su tutte le condizioni operative del programma
- Caso degli input estremi (vettori di dimensione 0 o 1, variabili nulle)

Documentazione

- Inserimento di commenti esplicativi nelle varie parti del programma per facilitarne la comprensione dopo molto tempo dalla sua stesura o per terze persone

Manutenzione

- Modifica del programma per soddisfare il cambiamento delle specifiche con cui deve operare

Commenti

- L'importanza dei commenti e della documentazione:
  - i programmi vengono utilizzati più volte nel corso di tempi lunghi (mesi, anni)
    - fare cambiamenti (aggiunta di caratteristiche)
    - risolvere errori
  - documentare il programma serve per rendere chiaro ed evidente lo scopo delle varie parti del codice

Commenti

- Si devono evitare commenti inutili
- Es:

```
'Set the variable x to 5
x=5
```

- Si deve evitare di inserirne troppo pochi
- Un buon modo per verificare il livello di commento è quello di leggere solo i commenti (e non il codice) ed ottenere una chiara idea di cosa fa un programma e di come lo fa

Commenti

- La sintassi per aggiungere commenti in VBA è

```
' commento
```

- Tutto ciò che segue un apostrofo è considerato un commento ed è ignorato dal VBA

- Esempio:

```
' Dichiarazioni
Dim Nome as String
name = ActiveDocument.Name 'Acquisisci nome
```

Continuazione delle linee

- Può essere più conveniente per la lettura da parte di un essere umano spezzare su più linee il codice troppo lungo
- Lo si fa con un carattere di underscore _ preceduto da uno spazio e andando a capo subito dopo
- Esempio:

```
ActiveDocument.Paragraphs(1).Alignment = _
wdAlignParagraphCenter
```

Le costanti

- Vi sono due tipi di costanti
  - costanti letterali
  - costanti simboliche
- Le costanti letterali sono valori specifici, come numeri, date, stringhe che non possono essere modificate e che sono usate esattamente come vengono scritte
- Esempio:

```
num=45
nome="Mario Rossi"
oggi=#1/5/03#
```

Le costanti

- Le costanti simboliche fanno invece uso di un nome per riferirsi alla costante letterale
- per indicare che un nome si riferisce ad una costante si premette al nome l'identificativo Const
- Esempio:

```
Const nome = "Giovanni Verdi"
```

Le costanti

- E' buona norma dichiarare tutte le costanti simboliche all'inizio della procedura in cui sono utilizzate
- questo per migliorare la leggibilità

Enumerazioni

- Quando vi è un insieme ampio di costanti numeriche intere si può utilizzare il costrutto enum
- la sintassi è:

```
Enum nome_complessivo
si=0
no=1
forse=2
End Enum
```

Enumerazioni

- Esempio:

```
If risposta = si Then …
```

- Che è meglio di

```
If risposta = 0 Then …
```

- Un caso comune è di usare il costrutto enum per definire i colori in VBA:

```
Enum ColorConstants
vbBlue=16711680
vbRed=255
End Enum
```

Variabili e tipi di dati

- Una variabile è una scatola (una locazione di memoria) che può contenere valori di tipo specificato
- Il valore contenuto può variare durante il programma, da cui il nome "variabile"

Tipi di dato

- Vi sono 5 tipi di dato predefiniti in VBA:

Tipo	Dimensione	Intervallo
Byte	1	0-255
Boolean	2	true/false
Integer	2	-32768 +32767
Long	4	2 miliardi
Single	4	-3.4E38 +3.4E38
Double	8	-1.8E308 4.9E324
Currency	8	1E16
Date	8	1/1/100 12/31/9999
Object	4	
String	variabile	
Variant	16	

La dichiarazione di variabili

- Dichiarare una variabile significa definirne il nome ed il tipo
- Le variabili si dichiarano nel seguente modo

```
Dim NomeVariabile As TipoDato
```

- ovvero premettendo lo specificatore Dim al nome della variabile e facendo seguire As e il tipo di dato

Nomi per variabili

- Qualsiasi carattere alfabetico (maiuscolo e minuscolo) e il segno di underscore _ vanno bene per formare il nome di una variabile
- si possono usare anche numeri a patto che non siano i primi caratteri del nome
- Esempi corretti:
  - nome_cognome
  - nome23
  - NomeCognome
- Esempi non corretti:
  - 23nome
  - nome%cognome

Nota

- Se non dichiarate il tipo VBA assumerà che il tipo sia Variant
- si possono mettere più dichiarazioni su una stessa linea come in

```
Dim a As Integer, b As Integer, d As Double
```

- non si possono dichiarare con

```
Dim a, b As Integer, d As Double
```

- cioè ogni variabile deve avere specificato il tipo

Variabili implicite

!! Attenzione !!

- Purtroppo il Basic permette di utilizzare variabili senza che siano precedentemente dichiarate!
- Una variabile di questo tipo viene considerata Variant e dichiarata automaticamente ed implicitamente quando viene usata
- questo è un pessimo stile di programmazione
- e' buona norma dichiarare tutte le variabili con il loro tipo all'inizio del programma che le usa

Nota

- E' facile vedere come questo possa creare errori difficili da individuare
- si pensi ad un programma dove indichiamo una variabile che debba contenere il risultato di un calcolo con il nome ilMioRisultato e per sbaglio utilizziamo successivamente la variabile indicandola con il nome llMioRisultato
- in questo caso il VBA istanzierà automaticamente una nuova variabile che non conterrà il dato che ci aspettiamo!

Dichiarazione esplicita

- Per rendere la dichiarazione delle variabili obbligatoria si premette al programma l'istruzione

```
Option Explicit
```

- questo fa in modo che venga segnalato come errore l'uso non dichiarato di una variabile

Tipo numerico

- I numeri possono essere manipolati e memorizzati tramite variabili di tipo Integer, Long, Single, Double e Currency
- In generale avremo a che fare solo con
  - Integer per trattare interi (positivi e negativi)
  - Single per trattare numeri decimali (positivi e negativi)

Tipo Booleano

- Il tipo Boolean prende solo due valori true e false
- è utile quando vogliamo realizzare funzioni che possano verificare delle condizioni su proposizioni e determinarne il valore di verità

Tipo stringa

- Il tipo string è una sequenza di caratteri
- La stringa vuota non ha alcun carattere
- Una stringa può contenere lettere, numeri, segni di interpunzione
- ma può anche contenere caratteri speciali di controllo come ad es:
  - vbCrLf per andare a capo
  - vbTab per inserire un segno di tabulazione

Concatenazione di stringhe

- Per unire fra di loro una o più stringhe si usa l'operatore di concatenazione &
- Esempio:

```
Dim nome As String
Dim cognome As String
Dim ris As String
nome="Mario"
cognome="Rossi"
ris=nome & cognome
```

Tipo Data

- Permettono la manipolazione e la memorizzazione di dati di tipo Data
- Si possono assegnare tramite costanti letterali di data come in

```
Dim dt As Date
dt=#1/2/98#
```

- o in forma di stringa

```
dt = "January 12, 2001"
```

- Nota: noi non le useremo

Tipo Variant

- Il tipo Variant permette di memorizzare un qualsiasi altro tipo di dato
- in genere è uno spreco di memoria utilizzare i variant invece che il tipo appropriato
  - si pensi ad esempio ad un vettore di interi di grosse dimensioni
- inoltre mantenere l'informazione del tipo di dato associato con una variabile permette di evitare errori logici

Oggetti di Word

- Il VBA per Word fornisce un elevato numero di tipi di dato specifico per Word
- Fra questi:
  - Document
  - Font
  - Paragraph
  - Table
  - etc

Dichiarazione e assegnazione di Oggetti

- La dichiarazione di una variabile di un tipo oggetto avviene nel modo usuale tramite lo specificatore Dim

```
Dim doc As Document
Dim fnt As Font
```

- L'assegnazione invece avviene diversamente: si deve premettere lo specificatore Set

```
Set doc = ActiveDocument
```

Vettori

- Un vettore o array è una collezione di dati che sono manipolabili tramite uno stesso nome e un indice.
- Si dichiara nel modo seguente

```
Dim vet(1 To 100) As Integer
```

- oppure

```
Dim vet(99) As Integer
```

- infatti se si omette il primo indice questo viene considerato 0

Vettori dinamici

- Nel caso in cui non sia nota a priori la dimensione dei dati si può dichiarare un vettore come

```
Dim vect() As Integer
```

- dopodichè è possibile specificarne la dimensione utilizzando l'istruzione ReDim

```
ReDim vect(1 To 100)
```

Preserve

- Il ridimensionamento di un vettore comporta la distruzione dei dati contenuti
- così non sarebbe possibile ridimensionare un vettore via via che ne sorgesse la necessità durante l'esecuzione di un programma
- per conservare i dati si inserisce lo specificatore Preserve, ottenendo:

```
ReDim Preserve vect(1 To 200)
```

UBound

- Nel caso in cui si voglia conoscere la dimensione di un vettore dinamico si utilizza la funzione UBound che restituisce il limite superiore (Upper Bound) del vettore
- UBound(vect)
- Nota: un vettore di capacità 100 elementi può contenerne un numero minore: UBound restituisce la capacità del vettore e non quanti elementi effettivamente ci siano

La visibilità (scope)

- Le variabili (stesso identico discorso per le costanti) hanno una visibilità all'interno delle varie parti che copongono un programma
- ciò significa che le variabili possono essere dichiarate in un punto del programma ed usate in un altro
- quando una variabile non è più visibile non si può più fare riferimento ad essa

Visibilità

- Le variabili possono avere visibilità locale o globale
- hanno visibilità globale quelle variabili dichiarate al di fuori di ogni procedura
- hanno visibilità locale le variabili dichiarate all'interno di una procedura o funzione

La inizializzazione delle variabili

- Quando una procedura inizia la propria esecuzione tutte le variabili sono inizializzate con un valore di default (in genere 0 per i numeri e la stringa nulla per le stringhe)
- tuttavia è bene non affidarsi a queste inizializzazioni automatiche
- è buona norma inizializzare esplicitamente le tutte le variabili

Gli operatori

- Gli operatori sono dei simboli e parole chiave in VBA usati per rappresentare in modo più compatto istruzioni frequenti
- Gli operatori si dividono in:
  - operatori aritmetici
  - operatori di stringa
  - operatori logici
  - operatori comparativi

Operatori

- Operatori aritmetici
  - addizione +
  - sottrazione -
  - moltiplicazione *
  - divisione /
  - divisione intera \
  - esponente ^
  - modulo Mod

Operatori

- Operatori di stringa
  - concatenazione &
- Operatori logici
  - E AND
  - O OR
  - negazione NOT

Operatori

- Operatori comparativi
  - Eguaglianza =
  - Maggiore >
  - Minore <
  - Magg. eguale >= oppure =>
  - Minore eguale <= oppure =<
  - Non eguale <> oppure ><