LA PROGRAMMAZIONE

- Abbiamo visto come le informazione vengono codificate per potere essere elaborate dal computer.
- Ora passeremo a vedere gli strumenti che abbiamo a disposizione per programmare il computer, per fare in modo, cioè, che il computer elabori le informazione che gli forniamo in modo utile per noi.

UN PO' DI STORIA - GLI ANNI '40

- All'inizio, negli anni '40, l'unico metodo per programmare era il **linguaggio macchina**.
- Il lavoro del programmatore consisteva nel settare ogni singolo bit a 1 o 0 su enormi computer che occupavano stanze intere e pesavano decine di tonnellate.
- I monitor non esistevano; i dati e i programma si fornivano al computer su schede perforate e il computer mandava i risultati su telescriventi.
- In questo modo gli scienziati riuscirono in pochi mesi a completare i calcoli per costruire la prima boma atomica, calcoli che se fatti a mano, con calcolatrici meccaniche avrebbero richiesto anni.

GLI ANNI '50

- Viene progettato il **FORtrAN** (FORmula trANslator), il cui utilizzo era ed è prettamente quello di svolgere in maniera automatica calcoli matematici e scientifici,
- Viene progettato l'**ALGOL** (ALGOrithmic Language), altro linguaggio per applicazioni scientifiche sviluppato da Backus (l'inventore del FORtrAN) e da Naur.
- Backus e Naur mettono a punto un metodo per rappresentare le regole dei vari linguaggi di programmazione che stavano nascendo.

GLI ANNI '60

- Nel **1960** venne presentato il **COBOL** (COmmon Business Oriented Language), ideato per applicazioni nei campi dell'amministrazione e del commercio, per l'organizzazione dei dati e la manipolazione dei file.
- Nel **1964** fa la sua comparsa il **BASIC**, il linguaggio di programmazione per i principianti, che ha come caratteristica fondamentale quella di essere molto semplice e, infatti, diventa in pochi anni uno dei linguaggi più utilizzati al mondo.

GLI ANNI '70

- Intorno al 1970, però, Niklus Wirth propone il **PASCAL** per andare incontro alle esigenze di apprendimento dei neo-programmatori, introducendo però la possibilità di creare programmi più leggeri e comprensibili di quelli sviluppati in basic.
- Pochi anni più tardi fa la sua comparsa il **C**.
- C e Pascal segnano una svolta importante in quanto sono linguaggi strutturati: struttura dei dati, struttura del codice.

GLI ANNI '80

- Negli anni ottanta comincia ad affermarsi la Object Oriented Programming; la programmazione orientata agli oggetti basata sul nuovo concetto di Classe.
- Il primo linguaggio a proporla ai programmatori professionisti è il C++ .
- C e Pascal segnano una svolta importante in quanto sono linguaggi strutturati: struttura dei dati, struttura del codice.

JAVA

- È il primo linguaggio progettato appositamente per la programmazione orientata agli oggetti (non è cioè l'adattamento di un linguaggio preesistente.
- Il programma ottenuto non è un programma destinato a *girare* in una ambiente specifico (Windows o Macintosh o Linux) ma gira su un computer virtuale, la *Java Virtual Machine*. Basterà installare la versione specifica della *Java Virtual Machine* per il sistema operativo specifico e lo stesso programma potrà girare in ambienti completamente diversi.

LA PANORAMA ATTUALE

- L'enorme diffusione dei personal computer ha influito anche sul mercato dei linguaggi di programmazione, una volte sicuramente un mercato marginale.
- I produttori di software hanno tentato di assecondare questo processo cercando di offrire prodotti sempre più semplici da usare (Visual Basic, Visual C ++, Delphi, Visual Java, ecc. sono prodotti che rendono enormemente più semplice di un tempo sviluppare programmi nei moderni ambienti grafici a finestre
- Gli strumenti per lo sviluppo di contenuti multimediali on-line e off-line (pagine web, animazioni interattive, ecc.) si sono dotati di linguaggi di programmazione sempre più potenti.

LA PANORAMA ATTUALE

- Oggi il mercato offre molti prodotti.
- I linguaggi di programmazione, per loro natura, possono fare benissimo alcune cose e male altre.
- Per scegliere cosa usare bisogna prima decidere cosa si vuole produrre, qual è il nostro obbiettivo.
- Il nostro interesse è soprattutto rivolto allo sviluppo di contenuti multimediali. Può essere, comunque interessante dare un rapido sguardo agli strumenti che abbiamo a disposizioni per realizzare progetti orientati ad altri fini.

LA PANORAMA ATTUALE

- Per **Applicazioni Matematiche o di Ricerca** i linguaggi più adatti sono ancora i *vecchi* ***Fortran*** e ***Algol***.
- Per i **grandi progetti software** (sviluppo di sistemi operativi, applicativi complessi, applicazioni lato server) normalmente viene impiegato il ***C*** o il *C++.*
- Per lo sviluppo di **programmi gestionali** in ambiente Windows lo strumento più utilizzato è ***Visual Basic*** prodotto da Microsoft. Un'ottima alternativa al Visual Basic è, da qualche anno ***Delphi*** un prodotto Borland basato sul Pascal che consente anche il porting dei programma tra l'ambiente Windows e l'ambiente Unix/Linux grazie a *Kilyx* versione Unix di Delphi. Negli ambienti diversi da Windows si utilizzano C/C++ e ambienti di sviluppo dedicati (i cosiddetti ***CASE***) basati su C/C++.

APPLICAZIONI MULTIMEDIALI

- Dal punto vista del nostro corso meritano una particolare attenzione gli strumenti per lo sviluppo di applicazioni multimediali.
- Possiamo distingue due tipi principale di applicazioni multimediali: quelle **on-line** (pagine Web) e quelle **off-line**.

SVILUPPO DI PAGINE WEB

- L'attività di programmazione entra a due livelli nello sviluppo di contenuti da distribuire via Internet:
- **La formattazione delle pagine**
- **L'inserimento di oggetti programmabili**

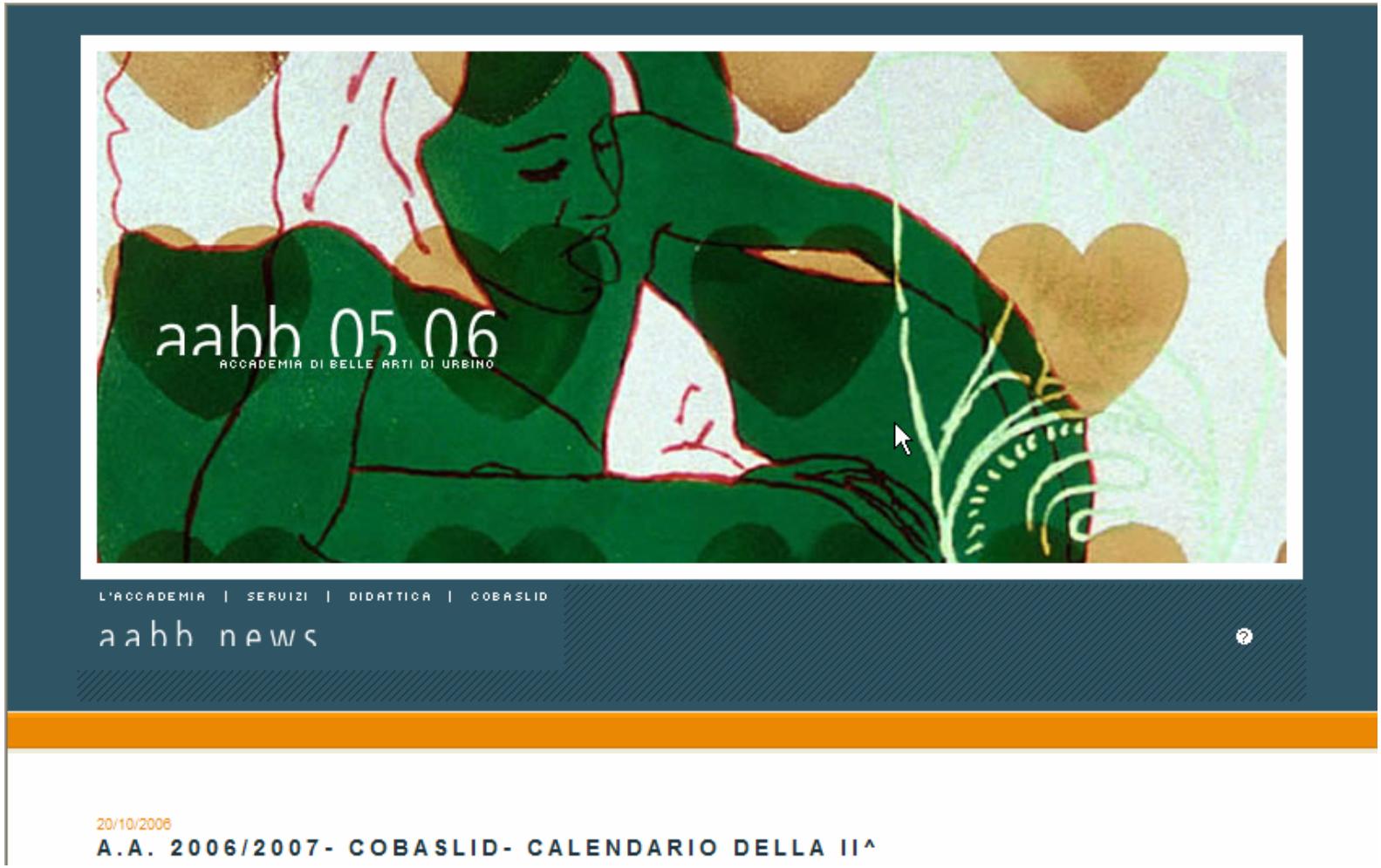
HTML (HYPETEXT MARKUP LANGUAGE)

- Descrive come una pagina web debba essere mostrata da un browser.
- È un linguaggio a marcatori: tutte le istruzioni proprie del linguaggio sono inserite tra due segni specifici (nel caso di HTML tra “<” e “>”) e costituiscono i cosiddetti **tag**.
- I browser cercano di interpretare i **tag** come istruzioni mentre il resto viene mostrato come testo.
- I **tag** che il browser non riesce ad interpretare vengono semplicemente ignorati.

HTML ESEMPIO DI CODICE...

```
<body onLoad="MM_preloadImages('images/nm_r1_c1_f2.gif','images/nm_r1_c2_f2.gif','images/
<table width="100%"  border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td align="left" valign="top" bgcolor="#2F5564"><br />
      <table width="753" height="301" border="0" align="center" cellpadding="10" cellspac
      <tr>
        <td><table width="474" height="272" border="0" align="center" cellpadding="0" cel
            <tr>
              <td align="center" valign="middle"><img src="images/rnd/14.jpg" width="733'
            </tr>
        </table></td>
      </tr>
    </table>
    <table width="757" border="0" align="center" cellpadding="0" cellspacing="0">
      <tr>
        <td><a href="accademia/" onmouseout="MM_swapImgRestore()" onmouseover=
"MM_swapImage('nm_r1_c1','','images/nm_r1_c1_f2.gif',1);"><img src="images/nm_r1_c1.gif"
"nm_r1_c1" width="92" height="18" border="0" id="nm_r1_c1" /></a></td>
        <td><a href="servizi/" onmouseout="MM_swapImgRestore()" onmouseover="MM_swapImage
><img src="images/nm_r1_c2.gif" alt="I servizi di segreteria, la biblioteca, il Forum" na
"0" id="nm_r1_c2" /></a></td>
        <td><a href="didattica" onmouseout="MM_swapImgRestore()" onmouseover=
"MM_swapImage('nm_r1_c3','','images/nm_r1_c3_f2.gif',1);"><img src="images/nm_r1_c3.gif"
```

...E RISULTATO



OGGETTI PROGRAMMABILI

- Specifici **tag** consentono di inserire nella pagine web vari tipi di componenti programmabili. I più importanti sono:
- **Script**
- **Applet**
- L'accoppiata **Object - Embed**

SCRIPT

- Il tag SCRIPT consente di inserire dei nuclei di codice che il browser è in grado di eseguire.
- I linguaggi a disposizione dello sviluppatore sono due: **VBSCRIPT** (che deriva dal Visual Basic e viene riconosciuto però solo da Internet Explorer) e **JAVASCRIPT** che deriva invece da JAVA ed è riconosciuto più o meno da tutti i browser.

SCRIPT ESEMPIO

```
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-
2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src; x.
src=a[i+2];}
}
//-->
</script>
```

APPLET

- Il tag **APPLET** consente di inserire in una pagina web un programma scritto in JAVA.
- Il programma dovrà rispettare alcune condizioni di sicurezza.
- Per girare il programma utilizza la **Java Virtual Machine** che deve essere installata sul computer.

APPLET ESEMPIO

```
<applet code="CellularAutomata3" archive="media/CellularAutomata3.jar" width="200" height="200">
</applet>
```

OBJECT-EMBED

- Il tag *OBJECT* (Internet Explorer su piattaforma Windows) insieme al tag *EMBED* (altri browser e Internet Explorer su Macintosh) consentono di inserire nelle pagine web oggetti di varia natura gestiti da estensioni dei browser o (in Windows) dai oggetti gestiti direttamente dal sistema operativo (i cosiddetti ActiveX).
- Qui segnaliamo due oggetti/plug-in che negli ultimi anno hanno avuto una grande diffusione: ***SHOCKWAVE FLASH*** e ***SHOCKWAVE DIRECTOR*** entrambi orientati alla sviluppo di applicazioni multimediali interattive, entrambi dotati di un potente linguaggio di programmazione.

OBJECT-EMBED ESEMPIO

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,0,0"
 WIDTH="100%" HEIGHT="100%" ALIGN="">
 <PARAM NAME=movie VALUE="Loader.swf">
 <PARAM NAME=menu VALUE=true>
 <PARAM NAME=quality VALUE=high>
 <PARAM NAME=bgcolor VALUE=#FFFFFF>
 <EMBED src="Loader.swf" menu=false quality=high bgcolor=#FFFFFF  WIDTH="100%" HEIGHT="100%"
ALIGN=""  TYPE="application/x-shockwave-flash" PLUGINSPAGE =
"http://www.macromedia.com/go/getflashplayer"></EMBED>
</OBJECT>
```

PAGINE STATICHE E DINAMICHE

- I tipi di pagine web di cui abbiamo fin qui parlato sono pagine statiche. Pagine, cioè, che vengono inviate al browser dal server web esattamente come sono state composte.
- Molti siti web oggi usano, invece, pagine dinamiche, pagine, cioè, il cui contenuto viene composto dal server al momento della richiesta sulla base dei parametri che gli vengono passati dal browser e del contenuto di database che il server può consultare.

STRUMENTI DI PROGRAMMAZIONE

- Questo tipo di programmazione, un tempo riservato agli specialisti, è oggi alla portata di qualsiasi programmatore.
- Gli strumenti più diffusi sono **PERL** e **PHP** per i server web che girano in ambiente Unix e Linux e **ASP** e la sua più recente evoluzione **ASP.NET** per Internet Information Server (il server web Microsoft).

APPLICAZIONI MULTIMEDIALI OFF-LINE

- Per quanto riguarda la Creazione di Giochi gli sviluppatori tendono a usare (per ottenere la massima efficienza) strumenti il più vicino possibile al linguaggio dei processori quindi **C++** e **Assembler**.
- Per altre applicazioni (didattiche, pubblicitarie, ecc) e giochi non troppo complessi vengono usati **FLASH**, **DIRECTOR** e **JAVA**.

ALGORITMI

FUNZIONAMENTO DEL CALCOLATORE

- Un computer per poter risolvere un qualsiasi problema ha bisogno di qualcosa che gli indichi esattamente cosa fare passo dopo passo
- Un computer si limita ad eseguire delle istruzioni "*macchina*" per l'esecuzione delle quali e' stato progettato
- Un'istruzione "*macchina*" e' un'istruzione "primitiva" comprensibile e direttamente eseguibile dal calcolatore senza bisogno di essere interpretata

SOFTWARE

- Qualsiasi compito per poter essere eseguito da un calcolatore deve essere tradotto in un insieme di istruzioni *macchina*
- Un software non e' altro che un insieme di istruzioni eseguibili dal calcolatore che nel loro insieme risolvono un problema o eseguono un determinato compito
- E' compito del programmatore "tradurre" un problema utilizzando solamente il set di istruzioni "primitive" comprensibili al calcolatore

ALGORITMO

Si può definire come un *procedimento* che consente di ***ottenere*** un dato ***risultato*** eseguendo, in un determinato ordine, un insieme di ***passi semplici*** corrispondenti ad azioni scelte solitamente da un insieme finito.

PROBLEMA

Problema:

Un contadino deve fare attraversare il fiume ad una capra, un cavolo, e un lupo, avendo a disposizione una barca in cui può trasportare solo uno dei tre alla volta. Se incustoditi, la capra mangerebbe il cavolo e il lupo mangerebbe la capra

SOLUZIONE

Algoritmo che descrive la soluzione:

1. Porta la capra sull'altra sponda
2. Porta il cavolo sull'altra sponda
3. Riporta la capra sulla sponda di partenza
4. Porta il lupo sull'altra sponda
5. Porta la capra sull'altra sponda

PROPRIETÀ FONDAMENTALI DEGLI ALGORITMI

- *Non-ambiguita'*: il procedimento deve essere interpretabile in modo univoco da chi lo deve eseguire (l'ordine di esecuzione dei passi deve essere non ambiguo)
- Eseguibilita': ogni istruzione dell'algoritmo deve poter essere eseguita senza ambiguità da un esecutore reale o ideale
- Finitezza: sia il numero di istruzioni che compongono l'algoritmo che il tempo di esecuzione dello stesso devono essere finiti

ALGORITMO BASE DEI CALCOLATORI

Finché non trovi un'istruzione di *halt* fai:

- Leggi un'istruzione dalla memoria
- Decodifica l'istruzione appena letta
- Esegui l'istruzione

RAPPRESENTAZIONE DI UN ALGORITMO

- Rappresentazione mediante **pseudolinguaggio** o **pseudocodice** che possiamo definire come un *linguaggio informale (ma non ambiguo) per la descrizione delle istruzioni.* Ogni riga rappresenta un'istruzione. Se non diversamente specificato, le righe si leggono dall' alto verso il basso.
- rappresentazione mediante **diagramma di flusso** (flowchart) che è una rappresentazione grafica in cui ogni istruzione è descritta all'interno di un riquadro e l'ordine di esecuzione delle istruzioni è indicato da frecce di flusso tra i riquadri.

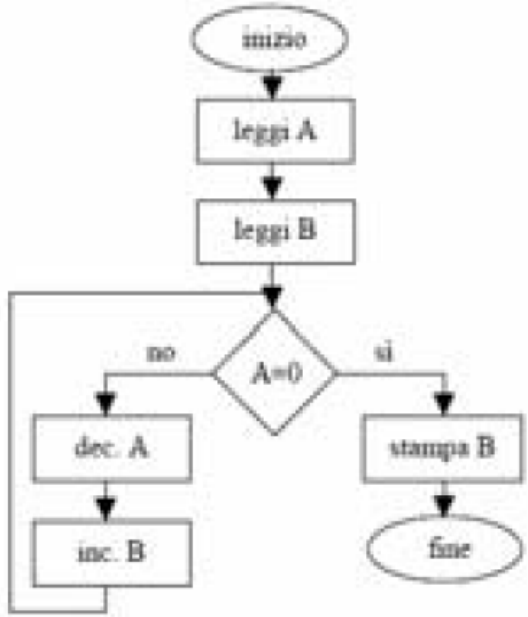
PROBLEMA:

sommare due numeri naturali utilizzando solo incrementi e decrementi

Pseudolinguaggio

1. Leggi A
2. Leggi B
3. Se A=0 vai all'istruzione 7
4. Decrementa A
5. Incrementa B
6. Vai all'istruzione 3
7. Stampa B

Diagramma di flusso



AZIONI PRIMITIVE

- La definizione di funzioni primitive permette di eliminare l'ambiguità

  Es: "Andare a lezione può essere irritante"

  - "Le lezioni causano irritazione"
  - "Il tragitto per arrivare dove si svolgono le lezioni e' irritante"

- Il livello di astrazione delle primitive deve essere scelto in funzione dell'esecutore

- L'insieme delle primitive più un set di regole per metterle insieme costituisce un *Linguaggio di programmazione*

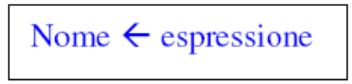
DEFINIZIONE DI UNO PSEUDOLINGUAGGIO

Istruzioni di inizio e di fine	**start**, **end**	Sono i punti d' ingresso e di uscita del flusso di esecuzione
Istruzione di assegnamento	nome←espressione	L' esecuzione di un' istruzione di assegnamento avviene in due fasi: la valutazione dell' espressione (utilizzando i valori correnti delle eventuali variabili che in essa compaiono) e l' aggiornamento del valore della variabile a sinistra del segno
Scelta tra due possibili attività	**if** (condizione) **then** (attività 1) **else** (attività 2)	Biforcazione del flusso in due percorsi alternativi (mutuamente esclusivi) la cui esecuzione è condizionata al verificarsi di una condizione. L' esecuzione di un' istruzione condizionale comporta innanzitutto la valutazione della condizione, e quindi l' esecuzione delle istruzioni che compongono uno dei due cammini.
Strutture iterative	**while** (condizione) **do** (azione)	Esecuzione ripetuta di una o più istruzioni. La ripetizione dipende dall' esito di un controllo. È fondamentale che l' esito del controllo dipenda da variabili il cui valore può essere modificato dall'esecuzione delle istruzioni del ciclo. In caso contrario il ciclo verrebbe eseguito o mai (risultando inutile) o all' infinito (violando la definizione di algoritmo). Le variabili da cui dipende il controllo sono dette variabili di controllo del ciclo. Ogni ciclo è composto da 4 elementi: l' inizializzazione delle variabili di controllo, il controllo della condizione da cui dipende l' iterazione, il corpo del ciclo (sequenza delle istruzioni da eseguire ciclicamente) e l' aggiornamento delle variabili di controllo.
Istruzione di salto	**goto** (riga di destinazione)	L'istruzione di salto sposta il flusso di esecuzione in un particolare punto dell'algoritmo. La riga di destinazione rappresenta, appunto, il punto in cui l' esecuzione riprenderà dopo il goto.
Definizione di procedure	**procedure** nome (istruzione 1 istruzione 2 istruzione 3)	Unità di programma utilizzabili attraverso invocazione da un qualsiasi punto del codice. Tutti i linguaggi di programmazione utilizzano questa strategia per rendere più leggibile il codice. Sinonimi di procedura sono: funzione, metodo, subroutine, sottoproramma ecc

DIAGRAMMA DI FLUSSO

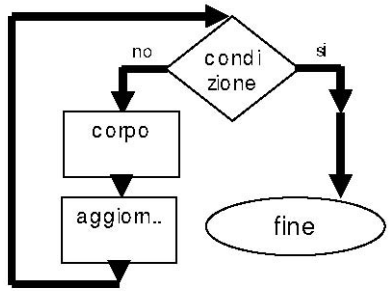
Inizio e fine

Assegnamento ed elaborazione



Scelta condizionata

Iterazione e salti



ESECUZIONE

Ecco come risulta l'agoritmo per sommare due numeri naturali utilizzando solo incrementi e decrementi:

```
1.  Start
2.  A ← input
3.  B ← input
4.  If(A=0) then (
5.          goto(9) )
6.  A ← A-1
7.  B ← B+1
8.  goto(4)
9.  Stampa B
10. end
```

GRANDEZZE

- *Costante*: quantità nota a priori il cui valore non dipende dai dati di ingresso e non cambia durante l'esecuzione (es: valore 0 nell'algoritmo precedente)
- *Variabile*: nome simbolico cui e' assegnato un valore che puo' dipendere dai dati di ingresso e variare durante l'esecuzuione (es: A e B)
- *Espressione*: sequenza di variabili, costanti o espressioni combinate tra loro mediante operatori (es: a+b*4)

COMPILAZIONE

Qualsiasi sia il linguaggio che abbiamo scelto il nostro codice dovrà essere tradotto, perché possa essere eseguito dal computer, in una serie di istruzioni che la CPU è in grado di interpretare ed eseguire. Questo processo, che chiameremo compilazione, a secondo dell'ambiente e del linguaggio scelto potrà avere diverse caratteristiche ma sarà comunque un passaggio necessario del nostro processo creativo.

IL PROCESSO

Se utilizziamo un diagramma di flusso per descrivere il processo di creazione di un'applicazione otterremo lo schema a fianco.