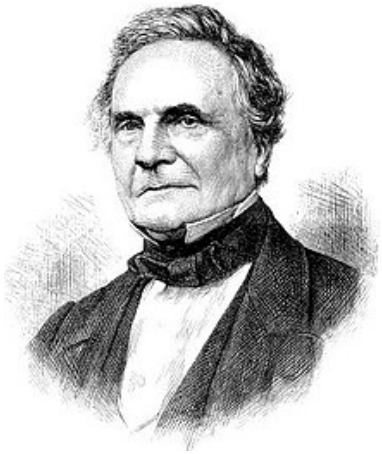


Charles Babbage

Ada Lovelace

Giovanni Antonio Amedeo Plana

Algoritmi e pseudocodifica
Classi seconde

Pablo Genova
gallini.genova@gmail.com

I. I. S. “Angelo Omodeo” **Mortara**

Indirizzo Tecnico-Economico **A. S. 2020 – 2021**

Approfondimento importante

Tenendo presente quanto visto l'anno scorso, quest'anno approfondiamo i seguenti aspetti, che l'anno scorso abbiamo trascurato:

- I **tipi** delle variabili (o delle costanti)
- Algoritmi **iterativi** (loop): come si fa un**'iterazione**?
- **Pseudocodifica:** le istruzioni che si danno effettivamente al computer (in forma semplificata rispetto alla **programmazione**)

Nella verifica saranno richiesti semplici algoritmi **iterativi** e bisognerà anche specificare i tipi delle variabili da introdurre e la **pseudocodifica** (limitatamente alle istruzioni che vedremo)

Chi sono quei personaggi della prima slide?

sono alcuni dei precursori dell'informatica

Augusta Ada Byron

(Londra 1815 – Londra 1852),

più nota come **Ada Lovelace**, era la figlia di **Lord Byron**, il famoso poeta e rivoluzionario britannico.

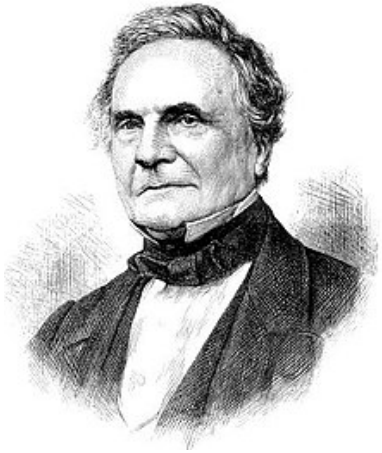
Ada lavorò insieme a **Charles Babbage** (vedi slide successiva), in particolare **descrisse e pubblicò un algoritmo per il calcolo dei numeri di Bernoulli** pensato per la macchina che Babbage voleva costruire.

Questo è il primo algoritmo della storia pensato specificatamente per l'implementazione in un computer (prima che esistessero i computer!!).

In suo onore negli anni 70 del Novecento è stato inventato il linguaggio di programmazione Ada tuttora utilizzato.

https://en.wikipedia.org/wiki/Ada_Lovelace#/media/File:Diagram_for_the_computation_of_Bernoulli_numbers.jpg

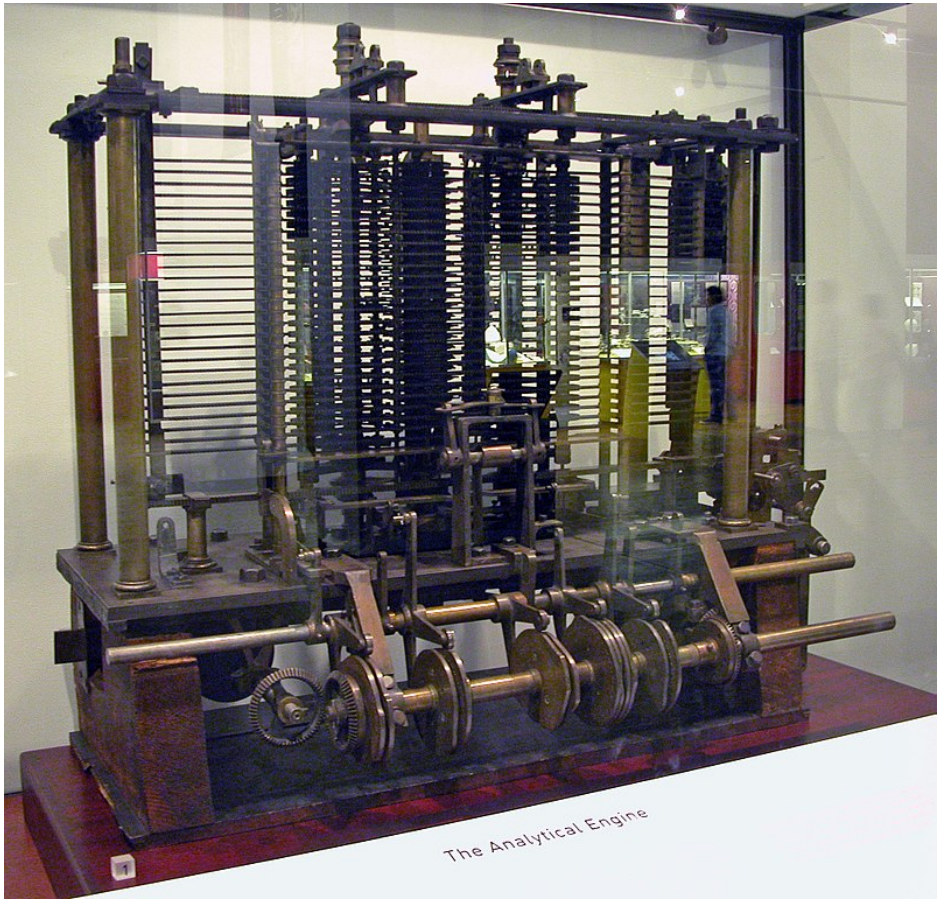
Charles Babbage



L'algoritmo di Ada era pensato per essere utilizzato nelle macchine che Babbage stava progettando.

Charles Babbage (Londra 1791 – Londra 1871) è da molti considerato il "padre del computer" perché inventò il primo computer, di tipo **meccanico**, detto la

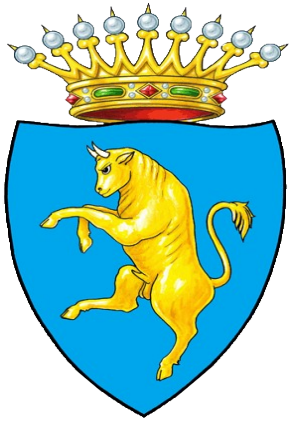
Macchina Analitica / Analitical Engine



Molte delle macchine di Babbage **NON** furono **MAI** costruite perché la tecnologia ottocentesca (non esisteva l'elettronica!) non permetteva la realizzazione di calcolatori come quelli attuali.
Ma le **idee** di Babbage erano corrette → ha anticipato i tempi!

Esempio di **macchina analitica** di Babbage al Museo della Scienza di Londra

E in Italia?



Torino

Non tutti sanno che … Babbage nel 1842 presentò a un seminario sulla **macchina analitica**, su invito di **Luigi Federico Menabrea** (1809 – 1896), ingegnere, generale e politico italiano

Giovanni Antonio Amedeo Plana (Voghera 1781 – Torino 1864) a cui è dedicata una scuola media a Voghera, nel 1831 costruì il **Calendario Meccanico Universale**

Con un insieme di ruote dentate, catene, viti e rulli girevoli, il Calendario di Plana è in grado di identificare un giorno qualsiasi dall'anno 1 all'anno 4000 oltre a fornire indicazioni sulla posizione della Luna e sulle maree.

Anche il Calendario di Plana si può considerare un **precursore meccanico del computer**

E' conservato presso la Sagrestia della Cappella dei Mercanti di Torino

https://it.wikipedia.org/wiki/Cappella_dei_Mercanti_di_Torino

Ripasso → Cos'è un algoritmo?

Algoritmo: **procedimento** che risolve un dato problema in un numero finito di passi

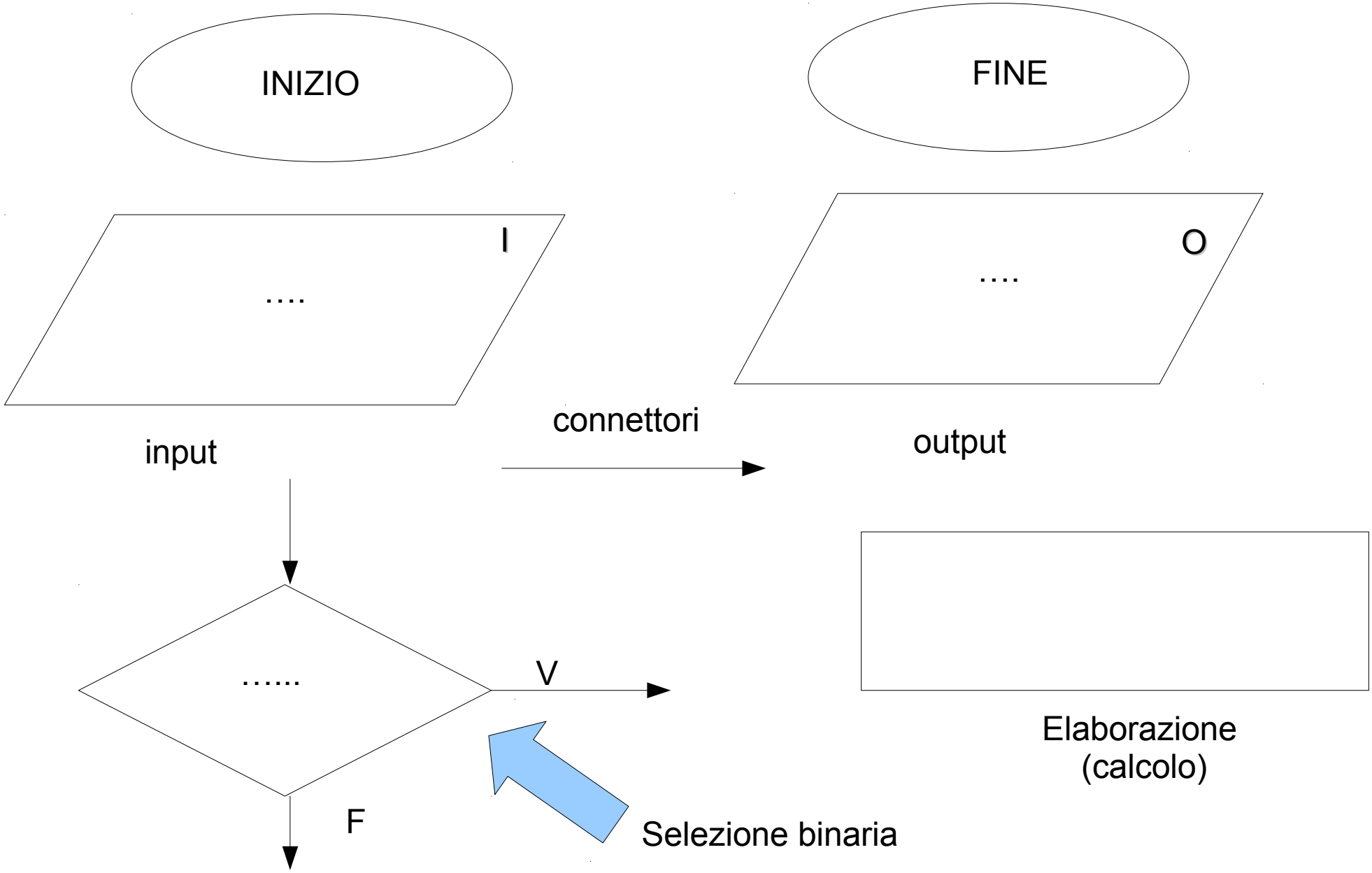
La proprietà di finitezza è molto importante (l'algoritmo deve finire!) inoltre tipicamente un algoritmo è

- non ambiguo,
- porta ad un risultato univoco,
- viene eseguito in un tempo finito
- i passi costituenti sono elementari (non ulteriormente scomponibili)

il passo dell'algoritmo si dice STEP in inglese

Curiosità: la parola algoritmo deriva da Muhammad ibn Mūsā al-Khwārizmī, matematico persiano della Corasmia attualmente Xiva (Хива) Uzbekistan

Ripasso → Blocchi elementari



Ripasso → Böhm - Jacopini

I due informatici italiani Corrado Böhm e Giuseppe Jacopini hanno dimostrato che

OGNI ALGORITMO PUO' ESSERE IMPLEMENTATO UTILIZZANDO LE SOLE TRE STRUTTURE:

SEQUENZA, SELEZIONE ED ITERAZIONE

Teorema di Böhm-Jacopini

Approfondiremo questo aspetto: l'ITERAZIONE

SEQUENZA E SELEZIONE le abbiamo già viste
(le sequenze sono i blocchi di elaborazione)

L'ITERAZIONE E' UNA STRUTTURA CHE PERMETTE DI RIPETERE un dato numero di volte una o più operazioni

L'ITERAZIONE è anche nota come CICLO O LOOP

Ogni dato ha un suo tipo

Ogni dato, sia esso costante o variabile, ha un suo **tipo** (tipo di dato in inglese *data type*) che è un nome che specifica a quale insieme di valori appartiene quel dato.
In altre parole il tipo è il **formato** del dato per esempio tipo **INT** per indicare un numero intero, tipo **CHAR** per indicare un carattere.

In generale la definizione dei tipi dipende dal linguaggio di progammazione

Esempio di tipi (tratto da **SQL** il linguaggio usato nei database):

BIT	per un tipo booleano 1, 0 (TRUE/FALSE)
INT o INTEGER	per un intero (da -2147483648 a 2147483648)
REAL	per un numero reale (da 1E-38 a 1E38)
CHAR o CHARACTER	per un carattere (ad es 'o', 'P' '0')
DATE	per una data (formato “anno/mm/gg”)
TIME	per un tempo (formato “anno/mm/gg h:min:s:ms)

ce ne sono anche altri per esempio in C c'è **float** per la **precisione singola** e **double** per la **precisione doppia** (il numero reale è infinito, il computer deve approssimarlo con un certo numero finito di cifre nel tipo **double** ne usa di più rispetto al **float**, di conseguenza il calcolo sarà più preciso)

VEDIAMO DEGLI ESEMPI

STRUTTURA DELL'ESEMPIO DI PSEUDOCODIFICA

pseudocodifica

La parte che il **computer** capisce, molto simile al vero codice di programmazione.

Separiamo le linee di codice con il simbolo

;

vedi slide seguente

commento

commento **per l'umano**, utile a capire cosa stiamo facendo

Mettiamo il commento preceduto dal simbolo

//

vedi slide seguente

Esempio di definizione di variabile

int x; // definisco la variabile intera x, il tipo di x è int cioè intero
x = 5; // assegno ad x il valore 5

char nome; //definisco la variabile carattere nome, il tipo di nome è
//char cioè carattere

nome="Pablo"; // assegno ad nome il valore Pablo (che è un insieme di
// caratteri anche detta **stringa**, una stringa di caratteri)

ATTENZIONE AGLI ERRORI!!

nome= 3; //errore!!! sto eguagliando un carattere ad un intero!!

x="Pablo"; // errore!!! sto eguagliando un int (numero intero) ad un
carattere

E' IMPORTANTE DEFINIRE ED USARE IL TIPO GIUSTO

Se si sbaglia il tipo, se va "bene" c'è messaggio di errore e si blocca il programma, se va "male" il programma dà un risultato ASSURDO

Esempio 2 con i float

float x,y,z ; // definisco le tre variabili float x, y, e z, il loro tipo è float

x = 5.345; // assegno ad x il valore 5.345
y = 6.235; // assegno ad y il valore 6.235

z = x+y ; // assegno a z la somma di x e y

Quanto varrà z?

se la stampo (con opportuno comando) otterrò 11.58

ATTENZIONE AGLI ERRORI!!

float x,y ; // definisco le due variabili float x, y
int z; // definisco un intero z;

x = 5.345; // assegno ad x il valore 5.345
y = 6.235; // assegno ad y il valore 6.235
z = x+y ; // assegno a z la somma di x e y, **ma z è int!!**

Quanto varrà z?

se la stampo (con opportuno comando) otterrò o un messaggio di errore oppure 11 (troncamento delle cifre decimali)... risultato **sbagliato**

Esempio di iterazione Somma da 1 a 10

pseudocodifica

int S = 0 ;

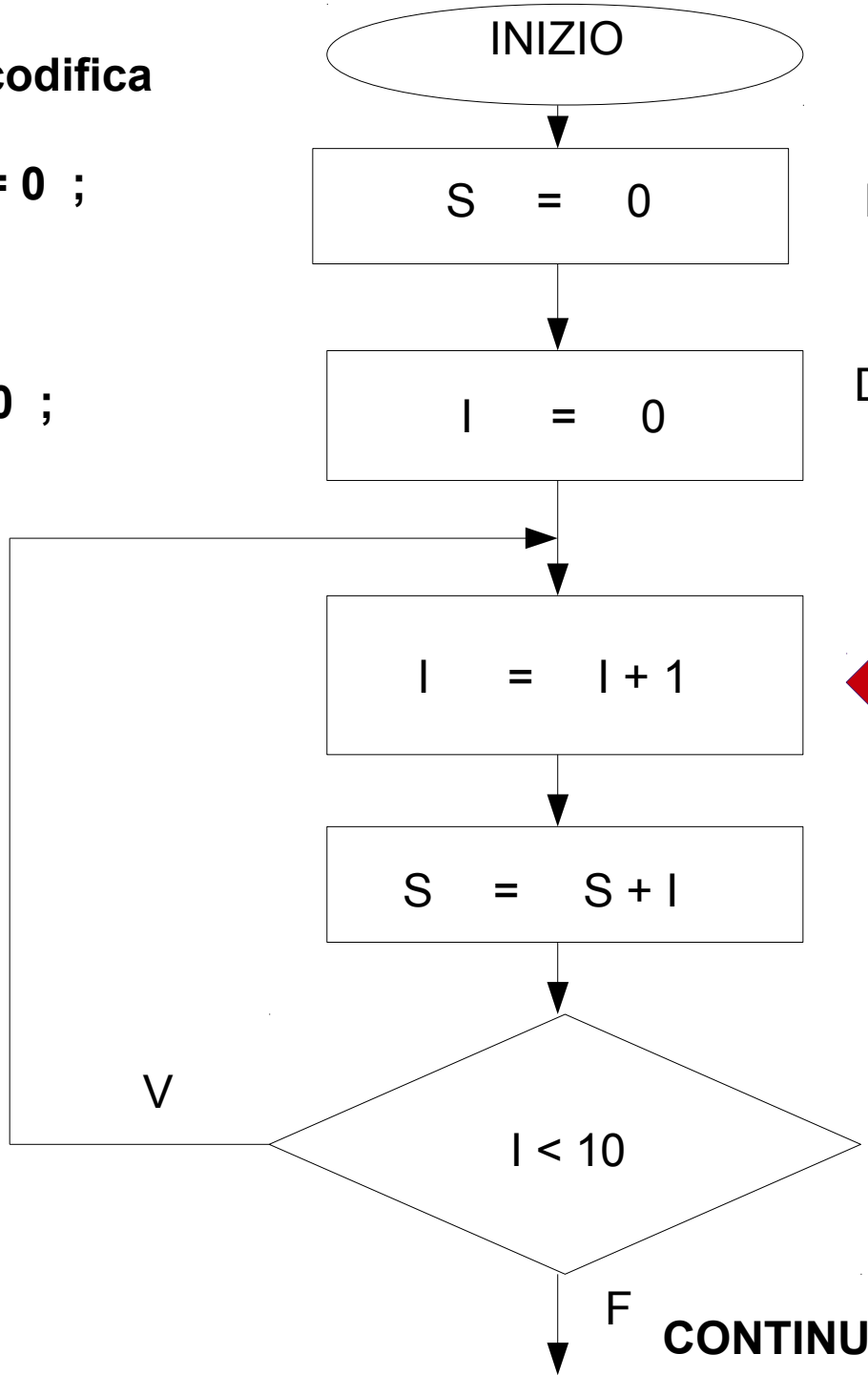
int I = 0 ;

I = I + 1 ;

S = S + I ;

while I < 10;

finché I<10;



Definisco la variabile S e la metto a 0

Definisco la variabile I e la metto a 0
I è un contatore

Incremento di 1 il contatore
metto I+1 in I
Questo serve a fare il CICLO

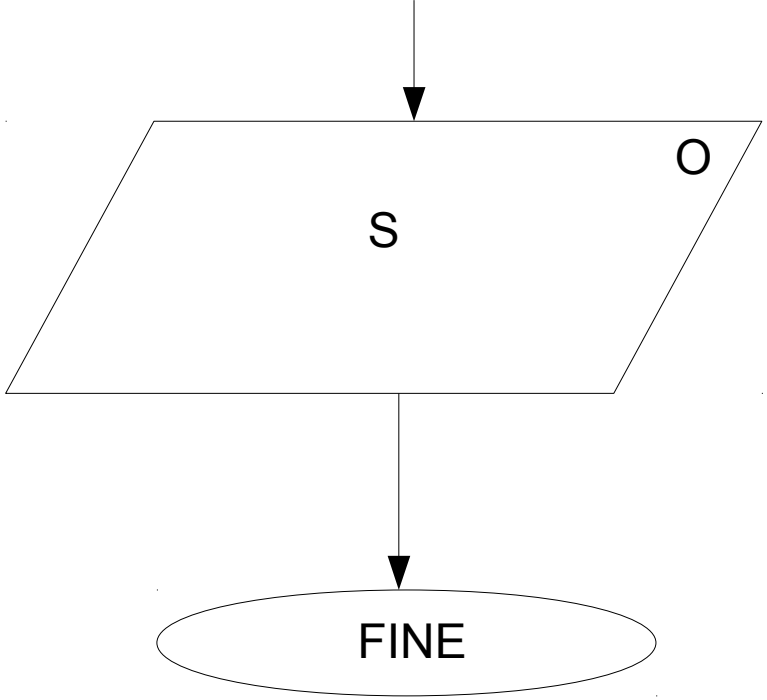
Incremento la somma di I, questo serve a fare la somma

BLOCCO IL CICLO QUANDO HO RAGGIUNTO IL NUMERO RICHIESTO (10)
attenzione è minore **stretto**
è < NON ≤

CONTINUA IN SLIDE SUCCESSIVA

INIZIA IN SLIDE PRECEDENTE

print S ;
stampa S ;



VISUALIZZO S

I cicli sono in qualche modo l'essenza della programmazione perché vi permettono di dire al calcolatore di ripetere <u>tante volte</u> una data operazione e quindi di fare in poco tempo calcoli <u>complessi o lunghi</u> per l'umano.

CICLO FOR E CICLO WHILE

Il diagramma precedente esemplifica un classico problema di iterazione, si noti che NON è l'unico modo di risolvere il problema.

La **pseudocodifica** (le istruzioni a fianco dei diagrammi di blocco) è molto simile al codice che il programmatore scrive per far sì che il computer crei il programma.
Le istruzioni precise dipendono dal **linguaggio di programmazione** concretamente scelto ad esempio FORTRAN, C, C++, Java, Basic, Pascal, SQL, Python e tantissimi altri (spesso a seconda del tipo di problema c'è un programma “specializzato” nel risolverlo)

Nella pratica i cicli più comunemente usati sono di due tipi:

1) Il ciclo **for**
2) Il ciclo **while**

Vediamo la loro (pseudo)codifica

CICLO FOR

Utilizzo: **di solito è utilizzato quando si deve iterare un numero noto di volte,** in altre parole si sa già che si deve contare per ad esempio 10, 100, 1000 volte e si ripete l'operazione per quel numero di volte

Esempio di codice C++ usando il FOR:

```
int I  = 0 ;                    // assegno alla variabile intera I il valore 0

int S = 0;                  // assegno alla variabile intera S il valore 0

for (I = 0; I <= 10; I ++)   // questo è il ciclo for che va da 0 a 10 (incluso)
                              // il simbolo ++ indica l'incremento del contatore I
{                             // apro le parentesi graffe

S = S + I ;                  // eseguo l'istruzione di somma (sarà ripetuta 11 volte)

}                             // chiudo le parentesi graffe

cout<<S;                     // comando per stampare la somma (in C++)
```

Se sostituisco 10 → 1000000 faccio il ciclo un milione di volte!! wow ;-)

CICLO WHILE

Utilizzo: **di solito è utilizzato quando si deve iterare un numero non noto di volte,** in altre parole NON si sa QUANTE volte si deve iterare (dipende dal calcolo Stesso)

Esempio: **fino a quale numero intero** devo sommare in modo tale che la somma da 0 a quel numero sia minore o uguale a 10?

Ecco come risolverlo in C++ usando il WHILE:

```
int I  = 0 ;            // assegno alla variabile intera I il valore 0

int S = 0;              // assegno alla variabile intera S il valore 0

while(S<10)   // esegui l'istruzione finché S (la somma) vale 10 ciclo while

{                       // apro parentesi graffe

I = I + 1   ;           // incremento il contatore
S = S + I ;             // eseguo l'istruzione di somma

}                       // chiudo parentesi graffe

cout<<I;    // comando per stampare il numero I  ovvero il contatore  (in C++)
```

0 + 1 + 2 + 3 + 4 = 10

Confronta il while col for e trova le differenze ;-)

Esempi e problemi (da saper fare)

- Ripasso dei diagrammi di flusso visti l'anno scorso
- Conoscere i principali tipi di variabili (slide 6)
- Saper scrivere, **in pseudocodifica**, semplici problemi matematici (vedi slides 8,9)
- Conoscere i diagrammi di flusso del ciclo **for**, applicandoli alla somma, alla media, al prodotto (slides 10,11)
- Conoscere la pseudocodifica del ciclo **for** e del ciclo **while** comprendendo le differenze tra i due metodi (slides 12,13,14)

Per provare ed approfondire

Esistono dei semplici tutorial on line che per esercitarsi nei vari linguaggi di progammazione ad esempio per il *C*++

https://www.tutorialspoint.com/cplusplus/

→ cliccare la finestra *try it*

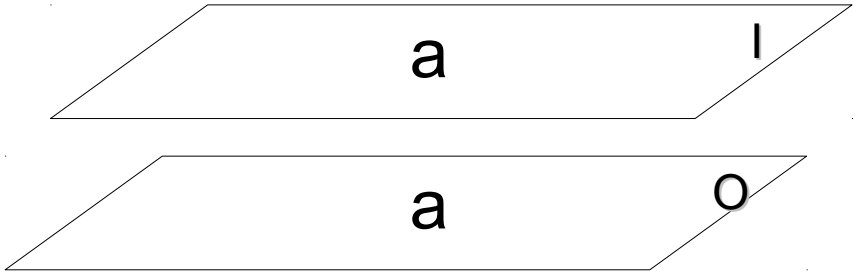
https://www.tutorialspoint.com/compile_cpp_online.php

→ questa versione mostra anche la shell linux

https://www.onlinegdb.com/online_c++_compiler

Buon divertimento ;-)

I comandi C++ : input / output



cin >> a INPUT

cout << a OUTPUT

Esempio in C++

```
#include <iostream>
using namespace std;

int main() {

int a; // definisco un intero

cout<<"inserisci un intero"<<endl;

cin>>a;// qui viene caricato l'intero

cout <<"il tuo intero è "<<a<<endl;

return 0;
}
```

In questo semplice esempio:

1) definiamo una variabile intera,

2) diciamo all'utente di digitarla su tastiera:
è un **output** → comando **cout**

3) la variabile viene inserita in memoria
→ è un **input** comando **cin**

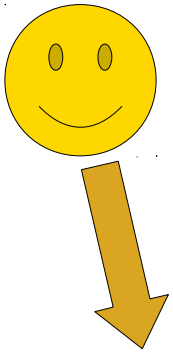
4) poi viene stampata nuovamente
→ comando **cout**

Osserva l'uso dei simboli >> e <<

I comandi C++ : i tipi int, float, double

Memorizza la corretta sintassi!

```
int num_0  ;   // definisco un intero  si può usare _ nel nome

num_0 = – 3; // int può essere un numero negativo

cout<<num_0;// lo stampo

float num_1; // definisco un float

num_1 =  7.3497; //lo assegno a 3,3497 → ci vuole il .

cout<<num_1<<endl; //lo stampo

double num_2; // definisco un double doppia precisione

num_2 = 5.653231123432; // qui ho bisogno di tante cifre decimali

cout<<num_2<<endl; //lo stampo
```

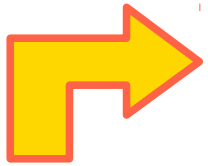


Questi (int, float, double) sono i tipi numerici più usati

Ricorda che il nome una variabile NON può iniziare con un numero int 2p; // ERRORE!!!
int p2; // OK

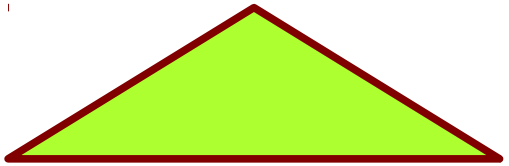
I comandi C++ : il tipo char

Memorizza la corretta sintassi! Il char è un po' più scomodo



la virgola **,** nella dichiarazione permette di definire più variabili dello **stesso tipo**, in questo caso **char**

```
char c1, c2, c3, c4, c5; //definisco 5 variabili char

c1 = 'c';    //assegno a c1 il carattere c ci vuole l'apice ' NON le "

c2 = 'i';    // assegno a c2 il carattere i

c3 = 'a' ; // assegno a c3 il carattere a

c4 = 'o' ; // assegno a c3 il carattere o

cout<<c1<<c2<<c3<<c4<<endl; //stampo il risultato
```



Attenzione che con questo metodo posso definire un solo carattere alla volta, per definire più caratteri con un solo comando occorre una stringa → vedi slide corrispondente

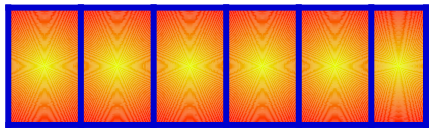
I comandi C++ : i vettori

Definizione: un **vettore** o meglio un **array** in C e C++ è **una struttura di dati formata da un insieme di elementi contigui di dimensione fissa e dello stesso tipo**

Esempi

vettore di float

```
int vettore[4] = {4, 5, 3, -9, 2};// definisco un vettore di int da 0 a 3

cout<<vettore[0]<<endl;// stampo il primo elemento si conta da 0 (zero)

float vettoref[] = {10.0, 13.0, -3.7, 17.0};//si può fare anche così →
                                              dimensione automatica

cout<<vettoref[4]<<endl;// ERRORE il vettore va da 0 a 3

double vettored[] = {10.01, 13.04556, -3.799888,};//vettore di double

cout<<vettored[1]<<endl;//stampo il secondo elemento: 13.04556
```



N. B. Il vettore ha dimensione **FISSA** E TUTTI GLI ELEMENTI DEVONO ESSERE DELLO **STESSO TIPO** → **ARRAY MISTI NON SONO CONSENTITI**

I comandi C++ : le stringhe

Le stringhe sono vettori (array) di char ecco **due** metodi per definirle:

char stringa[**5**] = {'C', 'i', 'a', 'o', '\0'};//definisco una stringa di **4** caratteri → **\0** termina la stringa **\0** è la stringa **null** (nulla)

char stringa2[] ="Ciao";// si può fare anche così → **dimensione automatica**

cout<<stringa<<endl;//stampo **tutta** la stringa **Ciao**

cout<<stringa[**2**]<<endl;//stampo il **terzo** carattere, la **a** di **Ciao**

Fare **attenzione** alla sintassi

nel metodo 1. bisogna usare l'apice **'**,
nel metodo 2. bisogna usare le virgolette **"**
nel metodo 1. bisogna indicare la dimensione includendo la stringa null,
nel metodo 2. non occorre né dire la dimensione né terminare la stringa.

N. B. I vettori misti non sono consentiti → !!! **int** vettore={3,'C', 'i', 4.2 , "o" }; !!!

I comandi C++ : il ciclo for

Il comando **for** permette di fare una **iterazione** per un numero **noto** di volte

devo **già** sapere quando mi devo fermare

Esempi

```
int vettore2[100]; //dichiaro un vettore di int 100 elementi da 0 a 99

for(int k=0;k<100;k++) vettore2[k] =k; //uso un ciclo for per
                                        //caricare gli elementi

for(int i=0;i<100;i++) {  //se dentro l'iterazione ci sono più comandi aprire la {

//… altro codice

cout<<"l'elemento numero "<<i<<" del vettore è uguale a "<<v[i]<<endl;

} //chiudo la graffa del for
```

1) k è un contatore

2) k parte da 0 e arriva a 99

3) viene incrementato di 1

for(int k=0;k<100;k++)

capire il for

k++ è lo stesso che k=k+1