1

Capitolo B81
c++, primo approccio

Contenuti delle sezioni

a. linguaggio C++, motivazioni e caratteristiche generali p. 4
b. primi esempi di programmi in C++ p. 6
c. costanti e variabili; dichiarazioni e assegnazioni p. 11
d. arrays e stringhe p. 17
f. entrata e uscita p. 21

21 pagine

B810.01 Questo è il primo dei capitoli dedicati al linguaggio di programmazione procedurale C++ [B81, B82,B83, B84, B86].

La presenza di questo linguaggio nella *esposizione* ha tre motivazioni.

Innanzi tutto la sua potenzialità didattica: esso consente di implementare concretamente e di esaminare anche in dettaglio algoritmi in grado di sostenere la comprensione di svariate nozioni della matematica discreta e della analisi numerica di base.
In effetti alcuni programmi in C++ sono presenti nella *esposizione*, non solo nei cinque capitoli citati, ma anche instretta connessione con la trattazione di nozioni strettamente matematiche.

La seconda motivazione riguarda il fatto che C++ si può vedere come corrispondente, limitato ma effettivamente sperimentabile del linguaggio di una macchina sequenziale programmabile, ossia del genere di strumenti che qui viene utilizzato per introdurre la computabilità, cioè la nozione che nella *esposizione* ha un ruolo primario nella individuazione delle finalità e delle caratteristiche generali della matematica.

Inoltre questo linguaggio di programmazione risulta sufficiente per implementare le procedure di sostegno alla redazione di questa stessa *esposizione* che sono state presentate discorsivamente in 05.

B810.02 Conviene ricordare che C++ è una evoluzione del linguaggio C che amplia tutta la sua portata, ma presenta alcune differenze sintattiche rese opportune dal fatto che si serve di librerie standard molto più ampie e dall'essere in progressiva evoluzione.

In questo capitolo r nel successivo introdurremo solo una parte delle prestazioni del linguaggio C++ e a questo proposito si può parlare della ,presentazione di un sottolinguaggio di C++ che chiamiamo **c++**, termine che vuole denotare un sottoinsieme ridotto del linguaggio C++.
Questo linguaggio è quasi completamente contenuto nel linguaggio C; più precisamente la sua portata è contenuta nella portata di C, ma comprende alcune prestazioni che a rigore sono assenti dal linguaggio C, ma che possono essere ottenute anche con C, ma a costo di qualche complicazione.

Quindi facciamo riferimento al linguaggio C++ piuttosto che al più vicino linguaggio C per vari motivi pratici.

C++ è più sicuro grazie alla più forte tipicizzazione; è dotato di librerie più ricche e in continua evoluzione; rispetto a C il linguaggio C++ è più vicino ai problemi e meno legato alle esigenze della macchina, ma consente ancora di essere consapevole dei dettagli delle operazioni sulle quali si basano le esecuzioni e che sono in stretta corrispondenza con i dettagli dei ruoli delle entità elementari, delle operazioni e delle strutture della matematica discreta presentati nei primi capitoli del tomo B.

C++ consente anche commenti più elastici da collocare a destra della coppia di segni "\\"; facilita il controllo di letture e scritture di caratteri grazie ai più semplici comandi `cin >>` e `cout <<`; agevola il controllo della memoria temporanea grazie agli operatori `new` e `delete`; attraverso i meccanismi delle classi di oggetti apre la possibilità della programmazione per oggetti, tecnica impegnativa e non cotemplata dal c++, anche se possiede notevoli potenzialità per la implementazione di strutture matematiche e in particolare per elaborazioni su di figure geometriche.

B810.03 In questo capitolo B81 c++ viene introdotto limitandosi alle caratteristiche che consentono di richiedere elaborazioni sopra numeri naturali e stringhe mediante semplici piccoli programmi che non fanno uso di sottoprogrammi, cioè che sono costituiti da un unico modulo.

Le pagine che seguono vogliono anche introdurre i primi concetti sulla programmazione e sulle prestazioni del computer con considerazioni che cercano di essere coerenti con quelle svolte come prime motivazioni per le attività matematiche.

Con questo intendiamo sottolineare la naturale vicinanza tra le basi discrete della matematica e le basi del trattamento delle informazioni, vicinanza dovuta all'interesse di entrambe verso le attività di calcolo e delle conseguenti prospettive, ampie e ambiziose, riguardanti le possibilità di ottenere soluzioni condivisibili e di portata generale per problemi che si pongono nel mondo odierno.

Dopo questa introduzione e alcuni paragrafi che seguono, mediante piccoli programmi o mediante frammenti di programma, i cosiddetti **snippets**, risulterà possibile presentare le implementazioni di molti dei procedimenti costruttivi che costituiscono componenti dignificativi della *esposizione*.

B810.04 c++ consiste in un sottoinsieme piuttosto circoscritto del `linguaggio C++ (wi)` che intende costituire uno strumento di programmazione utilizzabile concretamente e piuttosto facilmente con qualcuno dei molti sistemi software per lo sviluppo del linguaggio C++ attualmente disponibili.

In questo capitolo c++ viene introdotto limitandosi alle caratteristiche che consentono di richiedere elaborazioni sopra numeri naturali e stringhe mediante semplici piccoli programmi che non fanno uso di sottoprogrammi, cioè che sono costituiti da un unico modulo. Nel capitolo successivo82 verranno introdotte con una certa completezza varie altre prestazioni di C++.
Il contenuto dei capitoli B81 e B82 riteniamo che fornisca al lettore degli strumenti che gli consentano di comprendere o di scrivere lui stesso dei programmi che costituiscono implementazioni di una buona varietà di procedimenti costruttivi della matematica discreta e dell'analisi numerica di base.

Questi due capitoli non pretendono invece di fornire una preparazione professionale alla programmazione nel linguaggio C++.

Le pagine che seguono e i successivi capitoli dedicati a C++ vogliono anche introdurre alcuni primi concetti sulla programmazione e sulle prestazioni del computer con considerazioni che cercano di essere coerenti con quelle svolte come prime motivazioni per le attività matematiche.

Con questo intendiamo sottolineare la naturale vicinanza tra le basi discrete della matematica e le basi del trattamento delle informazioni, vicinanza dovuta all'interesse di entrambe verso le attività

di calcolo e delle conseguenti prospettive, ampie e ambiziose, riguardanti le possibilità di ottenere soluzioni condivisibili e di portata generale per problemi che si pongono nel mondo odierno.

Dopo questi due capitoli dedicati a c++, grazie alla presenza di molti piccoli programmi e di frammenti di programma, i cosiddetti **snippets**, risulterà possibile in B83 le implementazioni di molti procedimenti costruttivi che costituiscono componenti dignificativi della *esposizione*.
Molti di questi programmi fanno parte della collezione degli strumenti che sono stati messi a punto per sostenere lo sviluppo dell'*esposizione*.

B81 a. linguaggio C++, motivazioni e caratteristiche generali

B81a.01 Prima di introdurre le prime nozioni sul linguaggio c++, ritorniamo con maggiori dettagli sulle tre motivazioni della sua presenza nella *esposizione*.

La prima motivazione si può considerare pratica.
Il linguaggio C++ consente di implementare concretamente algoritmi in grado di sostenere la comprensione di svariate nozioni della matematica discreta e della analisi numerica di base; alcune di queste implementazioni sono già nella *esposizione*.
Si tratta di un linguaggio ampiamente diffuso disponibile liberamente su tutti i computers che si possono trovare in un scuola e in molte case. Si può quindi prospettare realisticamente che un buon numero di studenti o di comuni persone interessate alle applicazioni della matematica scrivano o comunque si procurino piccoli programmi con i quali possono prendere visione di esempi particolari di oggetti o processi matematici dei quali che vogliono conoscere meglio dopo averne esaminata la intoduzione e le proprietà.

Questa motivazione di C++ risiede nella possibilità di rendere i programmi, i frammenti di programma e i sottoprogrammi presentati nelle pagine che seguono concretamente sperimentabili e modificabili attraverso i numerosi compilatori e ambienti di sviluppo per i linguaggi C e C++. Questo sarebbe più problematico se si fosse scelto un linguaggio meno praticato e non sarebbe direttamente attuabile se si fosse adottato qualche genere di `pseudocodice (wi)`, ossia qualche presentazione espressa in termini più discorsivi e più facilmente accostabili.

La disponibilità di manuali sui linguaggi C e C++, tra l'altro, ci consente di introdurre le prestazioni di c++ che ci sembrano più rilevanti per l'*esposizione*con discorsi concisi che evitano di soffermarsi su molti dettagli che riteniamo di interesse secondario.

Diamo molta importanza alla apertura della possibilità di sperimentare fattivamente l'esecuzione di calcoli concernenti costruzioni matematiche riguardanti configurazioni discrete.

La possibilità di sperimentare, oltre ad avere evidente valenza pratica, può essere un robusto supporto alla comprensione di molte nozioni matematiche, in particolare negli ambiti delle strutture discrete, dei meccanismi inferenziali, dei calcoli approssimati e dei procedimenti per le applicazioni della disciplina.

B81a.02 La seconda motivazione è dovuta al fatto che C++ si può vedere come corrispondente limitato ma effettivamente sperimentabile del linguaggio di una macchina sequenziale programmabile, ossia di un genere di strumenti formali che qui abbiamo introdotto alla base della computabilità, cioè della nozione che qui è stata trattata per fornire motivazione iniziale

Questa seconda motivazione sta nella apertura della possibilità di sperimentare fattivamente l'esecuzione di calcoli concernenti costruzioni matematiche.

La possibilità di sperimentare, oltre ad avere evidente valenza pratica, può essere un robusto supporto alla comprensione di molte nozioni matematiche, in particolare negli ambiti delle strutture discrete, dei meccanismi inferenziali, dei calcoli approssimati e dei procedimenti per le applicazioni della disciplina. della matematica.
Per questa ragione si intendono aggiungere alla attuale *esposizione* alcuni programmi di simulazione delle macchine sequenziali programmabili e delle macchine di Turing.
Questi programmi saranno presentati in versioni che si proccupano principalmente della leggibilità, anche a scapito dell'efficienza e della generalità.

B81a.03 La scelta di c++, oltre che rendere disponibili algoritmi effettivamente verificabili, intende favorire la costituzione di librerie di programmi che portino anche a considerare come problema culturale e di ampia influenza quello della disponibilità di raccolte di procedure.

Inoltre i programmi che presenteremo pensiamo servano a evidenziare varie considerazioni alle quali diamo molto peso come le seguenti.
Tutti i procedimenti costruttivi riguardano prevalentemente sequenze finite, a partire da quelle su interi e stringhe.
È significativo e utile esaminare le differenze tra alcune formule matematiche e le loro possibili implementazioni.

B81a.04 Terza motivazione: C++ si dimostra adatto a concretizzare le procedure per sostenere la redazione di questa stessa *esposizione* che sono presentate in 05 e la descrizione di questo linguaggio può contribuire anche alla comprensione delle motivazioni e delle caratteristiche di tali procedure.
Anche i programmi che implementano queste procedure si intendono mettere a disposizione del lettore.
Occorre aggiungere che questi programmi cercano anche di sostenere l'interesse verso lo sviluppo di molti altri prodotti software in grado di sostenere in altri modi la crescita di studi della matematica e delle discipline contigue che diano importanza alle loro conseguenze costruttive e alle sperimentazioni sulle loro proprietà.

Sulle notevoli prospettive in questa direzione sarà opportuno ritornare con più estese considerazioni.

B81a.05 È opportuno anche segnalare che le attività di implementazione di algoritmi di interesse per la matematica conducono naturalmente a porsi una vasta gamma di problemi che riguardano nozioni presenti da tempo in questa disciplina.

Un prima tema che emerge dalle implementazioni riguarda le relazioni che intercorrono tra molte nozioni presenti nelle argomentazioni della matematica e nozioni di uso comune nelle dicussioni sopra la programmazione, il trattamento dei dati e la soluzione effettiva di problemi di vasto interesse.

In particolare meritano di essere ben chiarite la relazione tra insieme finito e sequenza, la relazione tra funzione e sottoprogramma, la relazione tra trasformazione lineare e matrice espressa come array bidimensionale.

Tra i problemi di natura matematica emersi dalle attività di trattamento dei dati possiamo ricordare quelli riguardanti le valutazioni quantitative sopra alcune classiche famiglie di algoritmi (selezione, ordinamento, visita di strutture, ...) e le questioni sulla computabilità (complessità, simulazione, ...) che si trovano in stretta relazione con i fondamenti della matematica.

B81a.06 Dovrebbe essere naturale per chi si interessa di matematica osservare come le odierne apparecchiature elettroniche consentano di effettuare una vasta e crescente gamma di calcoli di interesse matematico.
Il problema della implementazione della matematica dovrebbe essere considerato di primaria importanza e dovrebbe essere affrontato con prospettive ampie e generali.

Un prima constatazione che conviene avere presente quando si pensa a macchine per elaborazioni automatiche dice che ogni dispositivo fisico, disponendo di risorse di memoria e di tempo finite può trattare solo un numero finito di entità matematiche; va anche osservato che questa finitezza è condizionata anche dal fatto che può fornire solo informazioni che devono essere trattate distintamente sia da esecutori umani che da esecutori artificiali, soprattutto quelli attribuibili all'elettronica digitale.

B81 b. primi esempi di programmi in C++

B81b.01 Procediamo comn la presentazione e il commento di una serie di programmi molto semplici che presentano una progressione graduale di prestazioni.
Cerchiamo di prendere in considerazione programmi che soddisfano esigenze di una certa concretezza, in modo da presentare anche una progressione di esigenze reali e i tipi di meccanismi che i programmi nel linguaggio che abbiamo scelto rendono disponibili per dare qualche tipo di soluzione ai problemi incontrati.
Cominciamo con programmi che riguardano solo una console costituita da una tastiera per l'immissione di richieste e di dispositivo di uscita per l'emissione delle risposte del computer; attualmante il dispositivo di uscita più comune è uno schermo, ma si potrebbe anche avere una stampante; nel passato spesso si interagiva con il computer con una telescivente.

B81b.02 Il primo tipo di programma che viene presentato da sostanzialmente tutte le introduzioni, si limita a fornire una informazione elementare.
Il nostro primo programma intende spiegare se stesso.

```
#include <iostream>
using namespace std;
int main() {
    cout << "Sono un programma in grado di scrivere una frase leggibile.";
    return 0; }
```

Nel testo sorgente precedente le prime due righe servono a rendere disponibili al programma dei meccanismi di utilità generale dei quali il programmatore si può limitare a conoscere gli effetti, evitando di entrare nei loro dettagli operativi.
Su questi meccanismi torneremo più avanti; per ora ci basta segnalare che la prima linea rende disponibili vari meccanismi per le operazioni di ingresso e uscita (I/O) e la seconda consente di servirsi di una estesa gamma di nomi.
Viene poi l'intestazione del programma, `int main()`, che vedremo comparire all'inizio di ogni programma ed essere simile a tutte le intestazioni delle unità operative del linguaggio chiamate "functions".
Vediamo poi il cosiddetto "corpo del programma" delimitato tra i due segni coniugati `{` e `}`.
Tale corpo presenta una prima frase, che comporta la emissione su quella che funge da console di uscita del sistema, oggi di solito uno schermo mentre nel passato era più spesso la stampante di una telescrivente.
In questa frase troviamo `cout` seguito da `<<` e da una stringa di caratteri visualizzabili che rappresentano se stessi delimitati da due occorrenze di `"`, seguita a sua volta dal carattere `;`. `cout` è un comando di scrittura reso disponibile dalla cosiddetta direttiva che compare nella prima linea: `<<` precede la scrittura che viene emessa che per il linguaggio costituisce una costante simbolica.
Il corpo del programma si conclude con la semplice frase `return` seguita dal carattere `;`: questa frase richiede la fine della esecuzione del programma.
Più completamente diciamo che `return` restituisce il controllo al sistema operativo del computer, ossia all'apparato software che si occupa del controllo generale delle azioni del computer e che ora ci basta considerare come il meccanismo che consente l'esecuzione dei programmi che vengono successivamente richiesti da uno o piò operatori.

Tra i possibili programmi quine abbiamo presentato uno che fa intervenire solo la console del sistema controllandola con la frase contenente `cout`.

Osserviamo anche la presenza di una linea vuota; essa ha il compito di rendere più chiara la comprensione del testo sorgente. Lo stesso scopo hanno le posizioni vuote che precedono le due frasi del corpo del programma .
In generale è opportuno che nella redazione del codice sorgente il programmatore si preoccupi della sua leggibilità disponendo opportunamente le sue frasi, ossia le sue porzioni che vengono terminate da una occorrenza di ";" ; anche su leggibilità e frasi avremo modo di ritornare.

B81b.03 Vediamo ora un programma in grado di fornire all'operatore che lo fa partire alcune informazioni che possono rivestire concreto interesse. JQ `#include <iostream>`

```
using namespace std;
int main() {
    cout << " unita' di misura fondamentali\n";
    cout << " grandezza nome simbolo\n";
    cout << " Intervallo di tempo secondo s\n";
    cout << " Lunghezza metro m\n";
    cout << " Massa chilogrammo kg\n";
    cout << " Intensita' di corrente ampere A\n"
    cout << " Intensita' luminosa candela cd\n";
    cout << " Quantita' di sostanza mole mol\n";
    return 0;
}
```

Anche questo programma si limita ad emettere delle scritture visualizzabili, ma presenta due cose nuove.

Viene compiuta una sequenza di azioni che porta alla visualizzazione di 9 linee non banali; suggerisce anche che il linguaggo consente di avere programmi con tante frasi che rorniscono informazioni che in alcune circostanze possono essere assai utili.

Dopo aver ricordato che un computer ha la possibilità di maneggiare grandissime quantità di dati, il precedente programmino può essere presentato per cominciare a prospettare che gli odierni sistemi elettronici ci possono aiutare fornendoci grandissime quantità di dati.

Se inoltre si accennano i progressi delle tecnologie della informazione e della comunicazione, si può rendere conto che le informazioni a gestione digitale possono essere inviate e scambiate in tempi brevissimi anche a moltissimi altri sistemi simili al nostro in tutto il mondo e un po' oltre ; inoltre le grandi quantità di dati possono essere riordinate in tanti modi e ulteriormente elaborate, ad esempio con procedimenti statistici e grafici.
Dunque gli strumenti elettronico digitali digitali possono contribuire enormemente alle nostre conoscenze e quindi alle nostre azioni.

B81b.04 Abbiamo la prospettiva della gestione efficiente di testi che possono essere migliaia o milioni di libri.
Si osserva che i libri, i testi sorgente dei programmi e i testi di tutti i generi vengono prodotti controllando non solo caratteri visualizzabili, ma anche comandi come il passaggio da una line alla successiva, il passaggio da una pagina all'altra e la organizzazione di tabelle.

Quindi i linguaggi di programmazione si sono attrezzati per trattare anche segni che riguardano azioni come le precedenti.
In effetti nelle frasi di scrittura viste sopra compare il digramma "`\n`" che conclude le scritture ed ha funzione di richiedere il passaggio alla linea successiva.
Questo è un esempio di "escape sequence", scrittura che si serve di più caratteri semplici per rappresentare segni non visualizzabili.
Altri esempi sono "`\t`" per ottenere una tabulazione orizzontale, e "`np`" per ottenere il passaggio a nuova pagina.

Se osserviamo le frasi di emissione precedenti si deduce che devono essere disponibili altre escape sequences. Evidentemente in una scrittura è necessario far comparire anche il carattere "`"`" utilizzato per delimitare le costanti simboliche e "\", backslash, utilizzato come segno iniziale delle stesse escape sequences.
In effetti per questi scopi C++ dispone, rispettivamente, delle escape sequences "`\"`" e "`\\`".

Per quanto riguarda la gestione automatica dei testi in generale ocorre segnalare che si tratta di una problematica complessa per la quale sono sviluppati strumenti molto complessi.
Qui ci occupiamo inannzi tutto dei testi sorgente nel linguaggio C++ e quindi affrontiamo solo marginalmente problemi come quello del controllo dei caratteri nelle diverse fonti tipografiche o il controllo delle immagini.
Presentando C++, come per molti altri linguaggi di programmazione, avremo quindi a che fare solo con alcune decine di caratteri.
Occorre tuttavia aggiungere che un linguaggio come C++è servito anche a costruire sistemi software in grado di trattare testi molto complessi e aa veicolarli attraverso canali informatici, per non parlare di immagini, animazioni e archivi di conoscenze di ogni genere.

B81b.05 Finora trattati solo dati simbolici, singoli caratteri o stringhe, costanti; sono informazioni che non cambiano e possono solo essere emesse.
Ora vogliamo trattare anche numeri sia costanti che modificabili grazie agli interventi di operazioni eseguite dai circuiti della unità centrale, il centro operativo del computer.

Partiamo seguente programma.

```
#include <iostream>
using namespace std;
int main() {
    cout << "sommiamo 12, il numero delle mele, e 17, il numero delle pere" << endl;
    cout << 17 + 12 << " frutti" << endl;
    cout << "area dei due rettangoli " << 20 * 26 + 22 * 35 << endl;
    cout << 45 - 63 << endl;
    return 0;
    }
```

Abbiamo quattro frasi operative, tutte doverosamente concluse da "`;`".
La prima fa emettere una stringa leggibile seguita dalla emissione della richiesta di passaggio a capo espressa da `endl` (sta per end line); dunque con una frase vengono emesse due dati, una stringa fissa e una richiesta tipografica (che equivale al precedente `\n` finale di stringa.
La seconda frase prende in considerazione due coatanti numeriche intere, richiede di sommarle e di emetterle; poi emette una stringa ed effettua un a capo.

La terza, dopo aver emessa una scrittura esplicativa iniziale, richiede il calcolo di una espressione numerica intera poco diversa da quella della matematica usule nota e quindi con un significato operativo ben intuibile e uquindi l'emissione del numero risultante seguito dalla richiesta di passaggio a nuova linea.
La quarta ci fa sapere che la sottrazioe viene rappresentata dal segno "-", fatto abbastanza prevedibile.

L'esecuzione del programma comporta dunque l'emissione delle linee

```
sommiamo 12, il numero delle mele, e 17, il numero delle pere
29 frutti
area dei due rettangoli 290
18
```

Questo esempio può essere facilmente generalizzato: si possono emettere i risultati del calcolo di più operazioni aritmetiche con operandi forniti da scritture di costanti numeriche alternati a scritture esplicative abbastanza chiare.

Possiamo quindi proporre vari esercizi.

Programma il calcolo del volume di un edificio a forma di cuboide largo 60 metri, profondo 16 e alto 27.

Scrivi un programma che riguardi un foglio rettangolare di carta per regali da 60 cm per 140 cm dal quale si vogliono ricavare tre fogli 50 per 40 per ricoprire tre scatole; il programma trova l'area della carta avanzata.

Scrivi un frammento di programma che emetta il peso in grammi di un lingotto con la forma di parallelepipedo retto con spigoli da 12 cm per 7 cm per 4 cm di una lega avente la densità di 7 kg al decimetro cubo.

B81b.06 C++, come tutti i linguaggi di programmazione per attività con esigenze quantitative (nellr scienze, nelle tecnologie, nelle costruzioni, nei commerci, nelle amministrazioni, ...) permettono di operare su numeri che si avvicinano ai numeri reali, le informazioni chiamate numeri in floating point.
Esempi di scritture di numeri reali costsnti
Esempi di calcoli di espressioni con costanti reali e intere.
Intervengono commenti

B81b.07 Si devono poter trattare informazioni derivate dai dati iniziali e da altri dati eper questi si devono usare le variabili.
Queste sono entità rappresentati da identificatori ai quali sono assoiatr celle-mm, celle di memoria, le quali ospitan valori che possono cambiare nel corso di una esecuzione.
Esempi i idntificatori.
I valori possono variare in insiemi diversi ed essere trattati in modi diversi dalle frasi del programma.
Si distinguono quindi tipi diversi
ppartSi devono distinguere i numeri fp dai numeri interi.
commenti vari di fine linea o su più linee successive
Cominciamo con due tipi di variabili

`int` - riguarda numeri interi come 123 o -45

`double` - riguarda numeri reali fp,come 19.25 or -9.98

Esempi di espressioni numeriche con numeri interi e reali tratti da geometria, fisiica elmentare e statistica elementare.

`char` - riguarda singoli caratteri visualizzabili come `'a'`, `'W'` o `'+'` o non visualizzabili forniti da scritture escape sequences.

`string` - riguarda sequenze di caratteri delimitate da doppi apici come `"string"`.

`bool` - prevede due valori vero o true e false o fasso
de dichiarare e da assegnare

B81b.08 secondo esempio e sua spiegazione semplice

B81b.09 altre semplici considerazioni

B81 c. costanti e variabili; dichiarazioni e assegnazioni

B81c.01 Per quanto detto, un programma per un computer dotato di un comune semplice corredo di unità periferiche (tastiera, stampante, monitor video, porta USB per dispositivi flash), può vedersi come un complesso di richieste espresse secondo regole formali piuttosto precise che, ogni volta che viene presentato al computer un adeguato complesso di dati di ingresso (numerici, logici, testuali, grafici, impulsi elettrici, ...) governa l'esecuzione di una sequenza di operazioni la quale, se non si verificano situazioni da considerare patologiche, fornisce nuovi dati con il ruolo dei risultati della elaborazione effettuata.

Un programma in ogni sua esecuzione elabora vari dati: oltre ai dati di ingresso, vi sono dati inseriti nel programma ed altri che il programma produce nel corso dell'esecuzione; in genere solo una parte di essi questi solo una parte viene emessa per costituire il risultato dell'esecuzione.

Tra i dati che un programma può elaborare nelle sue possibili esecuzioni vanno distinti quelli che in ciascuna esecuzione rimangono fissi da quelli che possono subire modifiche in conseguenza di qualche comando espresso nel programma; i primi sono detti **dati costanti**, i secondi **dati variabili**.

Un computer attuale, per effettuare le elaborazioni che ci si aspetta possano essergli richieste, deve essere dotato di norme operative che hanno la forma di programmi predisposti e che attualmente possono essere molto articolati, ma che l'utenteinteressato ai risultati deve conoscere soltanto in relazione alle sue motivazioni.
Una parte primaria degli strumenti in dotazione del computer costituisce il sistema operativo del computer (Windows, Unix, Mac OS, ...) , sistema di programmi che interagiscono con l'utente e danno tutti i supporti necessari alle manovre richieste per le molteplici prestazioni esecutive.

Per condurre efficacemente le esecuzioni governate dai programmi applicativi degli utenti scritti in un dato linguaggio il computer deve essere dotato anche di un sistema di strumenti (altri programmi predisposti) anch'essi decisamente complessi che chiamiamo **sistema di sviluppo**.

Qui ci limitiamo a segnalare solo due dei compiti di un sistema di sviluppo: (1) occuparsi della traduzione di un programma scritto nel linguaggio di programmazione in un complesso di istruzioni comprensibili dai dispositivi hardware e software costituenti il sistema di sviluppo stesso; (2) monitorare ciascuna delle esecuzioni dei programmi segnalando gli accadimenti che possono interessare l'utente e in particolare le situazioni che giudica erronee e le situazioni che fanno sospettare che l'elaborazione in corso non stia fornendo risultati corretti.

B81c.02 Anche il sistema di sviluppo di un linguaggio come C++ non deve necessariamente essere conosciuto in dettaglio da un programmatore; tuttavia è opportuno che egli abbia idee semplificate ma chiare delle sue componenti e delle sue prestazioni, in modo che possa prender facilmente le decisioni più opportuneper il raggiungimento dei suoi obiettivi.

La visione del sistema di sviluppo deve essere tanto più estesa quanto più è impegnativa e sistematica l'attività di programmazione e di utilizzo dei risultati.
L'approfondimento di questa visione da parte di un utente sistematico del computer si può configurare come formazione di un proprio metodo e di un proprio stile di lavoro.
Questa formazione sarà particolarmente impegnativa quando il computer viene utilizzato da intere squadre di programmatori che devono affrontare problemi computazionali impegnativi.

Ogni sistema di sviluppo di un linguaggio di ampia portata è un prodotto industriale decisamente evoluto che in buona parte presenta dipendenze dal computer utilizzato e dal sistema operativo istallato.

Esso svolge svariati compiti, soprattutto:
- tradurre le richieste formulate in ogni programma in richieste comprensibili per la macchina;
- individuare e segnalare opportunamente gli errori lessicali e sintattici che trova nel testo del programma e gli errori semantici potrebbe rilevare nella struttura del programma;
- segnalare le eventuali anomalie si verificano nel corso dell'esecuzione;
- mettere a disposizione ampie librerie di sottoprogrammi di utilità generale;
- supportare la gestione di programmi di sistema con i quali un programma scritto dall'utente può interagire

Dopo il precedente semplice accenno, ritorneremo su alcuni aspetti del sistema di sviluppo solo per chiarire questioni che si andranno ponendo nel corso della presentazione delle prestazioni del linguaggio.

B81c.03 Possiamo assumere che a ogni dato fisso o variabile che un programma elabora viene associata una cella di memoria destinata a contenere i valori che il dato viene ad assumere nelle successive fasi di una esecuzione del programma; per un dato variabile il valore registrato nella sua cella in una fase esecutiva viene detto **valore corrente** del dato.

La associazione di una cella a un dato viene effettuata dalla componente traduttore del sistema di sviluppo.

Senza entrare nelle tecniche adottate dal traduttore dei programmi in un dato linguaggio (che può essere diverso per i diversi sistemi operativi, possiamo pensare che il traduttore organizzi una tabella che associa a ogni dato che compare in un programma la corrispondente collocazione in una zona della memoria della macchina esecutrice.

La memoria del computer dal punto di vista di un programma si può considerare come un nastro nel quale si distinguono successive zone dedicate ad aggregati di dati (e quindi di celle) tendenzialmente omogenei appartenenti ai vari tipi (bytes per caratteri, celle per interi delle diverse taglie e altri che incontreremo) che vengono coinvolti dal programma.

B81c.04 Ciascuna delle celle-mm coinvolte da un programma viene individuata dall'indirizzo del primo dei bytes a essa assegnati e dal loro numero, corrispondente al suo tipo di dato.

In molte circostanze occorre saper controllare di una cella-mm sia l'indirizzo iniziale che la sua estensione; come vedremo il linguaggio C per questo offre diverse possibilità.

Chi formulava i programmi per i primi computers utilizzati poteva servirsi solo dei codici binari delle istruzioni della macchina e doveva servirsi degli indirizzi fisici delle celle-mm nelle quali collocare i dati da elaborare. Questo modo di lavorare veniva detto "programmazione nel linguaggio macchina" e risultava assai oneroso per la distanza tra codici delle istruzioni e indirizzi delle celle da una parte e operazioni da eseguire e nomi delle variabili dall'altra, a causa del gran numero di dettagli che il programmatore doveva avere presenti.

Un primo miglioramento si è avuto negli anni 1950 con la introduzione dei cosiddetti linguaggi simbolici di macchina; ciascuno di essi consentiva di controllare le prestazioni dei computers di un dato modello mediante simboli esprimenti istruzioni e mediante nomi per le celle-mm che il programmatore poteva scegliere in modo da poterli ricordare con facilità.

Successivamente gli sviluppi delle relative tecnologie hanno reso i computers sempre più miniaturizzati, efficienti, affidabili, versatili e diffusi e hanno indotto i produttori a dotarli di una varietà di dispositivi ausiliari rivolti a migliorare le interazioni uomo-macchina e i collegamenti tra la macchina e altre apparecchiature esterne, anche remote; tra queste ultime terminali per utilizzatori distanti, sensori, attuatori e altri computers impegnati in elaborazioni con finalità in comune.

B81c.05 Contemporaneamente e sinergicamente gli elaboratori elettronici sono stati dotati di una crescente varietà di strumenti software rivolti a facilitarne l'utilizzo: sistemi operativi, traduttori di linguaggi, librerie di sottoprogrammi, strumenti per la diagnosi di errori di programmazione e di malfunzionamenti dell'hardware, sistemi per la gestione di archivi di dati, strumenti per il sostegno allo sviluppo di programmi via via più ambiziosi, ...

L'utilizzo dei computers, le cui popolazioni sono passate dalle decine ai miliardi di esemplari, si è trasformato dall'essere una pratica guidata dalle esigenze di applicazioni specifiche e dalle caratteristiche di singole realizzazioni hardware al costituire una fenomenologia complessa da esaminare in relazione all'andamento dell'economia e delle imprese, allo sviluppo della società e a una nuova cultura che dà ampio credito alle metodologie e alle soluzioni condivise e alle esigenze di ampie categorie di utilizzatori più o meno diretti.

Tra queste ultime non si possono trascurare le categorie dei detentori di poteri riguardanti imprese finanziare, industriali, della logistica e dei commerci, dei media e dello spettacolo, i servizi sanitari e dell'istruzione, dei poteri militari e politici complessivi, nonché, indirettamente ma massicciamente la categoria dei consumatori di intrattenimento e videogiochi.

Naturalmente non ci addentreremo nei molteplici aspetti di questi della società odierna e ci poniamo dal punto di vista della adozione di un linguaggio di programmazione procedurale medio-alto e di portata vasta come il C++, cercando tuttavia di segnalare i collegamenti più influenti tra i suddetti sviluppi e le metodologie della programmazione.

B81c.06 In un linguaggio come C++ i dati costanti sono individuati mediante scritture che seguono precise convenzioni volte a esprimere accuratamente il loro significato, mentre le variabili sono identificate da nomi che dovrebbero essere scelti dal programmatore esaminando con accuratezza le possibili conseguenze sulle sue previsioni di lavoro.

È opportuno che la scelta degli identificatori delle variabili sia effettuata sull'intero complesso dei dati del problema preoccupandosi che essa faciliti il più possibile la individuazione ed il ricordo dei rispettivi significati e ruoli.
Questo conta tanto più quanto maggiore è la possibilità che il programma che sta scrivendo venga ripreso e ampliato o adattato in tempi successivi al suo primo utilizzp, soprattutto quando si prevede che su di esso intervengano nuove persone.

Il sistema di sviluppo del linguaggio si incarica di assegnare a tutti i dati in un programma le opportune celle-mm e di inserire i valori costanti nelle celle per i dati fissi. In genere anche nelle celle per i dati variabili sono inseriti dei valori iniziali prestabiliti, ma è dubbio che questo sia fatto in modo univoco sui tutti i computers e potrebbe essere fonte di rischi.
È quindi consigliabile che il programmatore si preoccupi del valore portato da ciascuna variabile prima di ogni suo utilizzo.

Serve adentrarsi nei dettagli dei collegamenti tra gli identificatori delle variabili e le celle-mm loro assegnate, dettagli che dipendono dal comportamento interno del sistema di sviluppo, solo per programmi per esigenze molto specifiche.
Per la massima parte dei programmi è sufficiente osservare che il sistema di sviluppo nella fase di traduzione del programma organizza una tabella che gli permette di conoscere questa relazione nelle fasi esecutive.
Una elaborazione che gestisce i dati servendosi di indirizzi per un nastro o per la memoria centrale, grazie alle tabelle di collegamento identificatori-indirizzi, risulta equivalente a una elaborazione che per i dati si serve di identificatori.

B81c.07 Veniamo alle regole per il controllo dei dati con il linguaggio C++.

Diciamo **alfabeto degli identificatori** l'insieme costituito dai caratteri alfabetici, dalle cifre decimali e dal segno "_".
Un identificatore si ottiene con una stringa di caratteri dell'alfabeto degli identificatori la cui iniziale non può essere una cifra ma deve essere una lettera o il segno "_".

Gli identificatori dei dati da elaborare sono introdotti in un programma da frasi dichiarative con le quali si stabilisce anche il tipo del dato identificato. Esempi:

```
boolean accettabile, presenza, daEsaminare;
char iniz, finale, categ;
integer j, k2r, lunInCar, valTot, maxValue, numPoints;
```

Veniamo alla scelta degli identificatori per un programma che prevedibilmente dovrà essere riadattato a varie nuove richieste applicative.
In linea di massima il programmatore deve scegliere tra due esigenze che possono entrare in conflitto: da un lato scegliere identificatori concisi, dall'altro decidere identificatori leggibili e mnemonici, cioè in grado di suggerire e/o ricordare il significato dei dati che l'identificatore va assumendo.
Negli esempi precedenti sono suggerite soluzioni intermedie con stringhe relativamente concise e abbastanza leggibili, in alcuni casi grazie al ricorso alle cosiddette "stringhe a dorso di cammello", stringhe scandibili in sottostringhe grazie alla comparsa di poche maiuscole entro le più numerose minuscole.

B81c.08 Vediamo come possono essere introdotte le scritture costanti dei tipi che ora ci limitiamo a considerare, cioè degli interi, dei valori booleani e dei caratteri visualizzabili. Queste scritture sono chiamate anche **literals**.

Le costanti intere possono essere espresse con notazioni posizionali relative a diverse basi.
Con notazioni decimali: 35 23009 -16 -1000000
Con notazioni ottali: `o14` sta per l'intero decimale 12 , `o1000` sta per 512.
Con notazioni esadecimali: `0xc` sta per 12, come l'equivalente scrittura `0XC`.

A ciascuna di queste costanti potrebbero essere assegnate celle-mm di estensioni diverse che ovviamente devono essere più estese per valori assoluti maggiori.

Un intero grande esige una cella-mm estesa, ma si potrebbe chiedere di inserire un intero piccolo in una cella-mm estesa.
Se ad esempio si vuole introdurre una costante intera da assegnare a una cella-64b, cioè se si vuole una costante del tipo chiamato `long integer`, è necessario usare una scrittura literal, ossia una notazione decimale seguita dalla lettera "`L`" o dalla "`l`": per esempio si predispone una cella-64b per il numero 12 scrivendo `12L`.

B81c.09 Sono disponibili le costanti booleane `TRUE` e `FALSE`, equivalenti rispettivamente ai bits 1 e 0.

Anche i contenuti dei singoli bytes possono essere introdotti con notazioni diverse costituenti i cosiddetti literals di carattere. Un tale literal è costituito da una rappresentazione del carattere da esprimere delimitata da da due segni di apostrofo, segno chiamato anche "single quote"' e "accento grave".

Tutti i caratteri sono rappresentabili con notazioni ottali di una delle forme `\`*o* `\`*oo* `\`*ooo* , dove *o* rappresenta una cifra ottale (una delle cifre da 0 a 7) e il numero espresso dopo il segno `\` rappresenta la posizione del carattere nella sequenza dei caratteri ASCII-8 [:d01].

I literals più semplici ed evidenti sono disponibili per i caratteri visualizzabili e si sono ottenuti dal carattere in causa delimitato da due segni di apostrofo : `'a'` `'!'` `'$'`.

Otto caratteri ASCII non visualizzabili importanti per la programmazione sono espressi mediante sequenze escape come segnalato dalla seguente tabella:

newline	hor.tab	vert.tab	backspace	carriage return	form feed	backslash	single quote
\n	\t	\v	\b	\r	\f	\\	\’
’\n	\t	\v	\b	\r	\f	\\	\”

B81c.10 Vediamo alcune frasi di dichiarazione e di assegnazione per variabili dei due tipi fondamentali dei numeri interi e dei bytes.

```
int step, stepNumMax;
short int matrIscritti, numIscritti, indScolari, numScolari;
long int numIntntAddr;
stepNumMax=200000000; numIscritti=102;
numScolari=28; numIntntAddr=2000000000;
```

La prima frase stabilisce che i primi due nomi identificano due variabili intere che occupano due celle-32b, cioè 2 bytes; la seconda dice che i 4 nomi che seguono la occorrenza di `short int`, stringa costituente una cosiddetta **scrittura chiave** o **scrittura riservata**, sono gli identificatori di altrettante variabili intere, ciascuna delle quali richiede una cella-16b; la terza richiede che `numIntntAddr` rinvii a una cella-64b in grado di contenere interi i cui valori possono variare nell'intervallo $[-2^{31} : 2^{31} - 1]$.

Le frasi precedenti sono dette **frasi dichiarative** ed hanno anche l'effetto di determinare gli indirizzi, ossia le posizioni nella memoria centrale, assegnati alle corrispondenti celle-mm e di predisporre che i rispettivi contenuti nel corso della esecuzion siano trattati come numeri interi con segno.

Le due ultime frasi assegnano i numeri scritti a destra del segno “=” come valori attuali delle variabili scritte alla sua sinistra.
Le frasi di questo tipo devono comparire dopo le frasi dichiarative delle variabili in causa (queste in genere collocate all'inizio di un programma) e devono comparire prima di ogni altra frase che coinvolga la rispettiva variabile.
Esse sono dette **frasi di inizializzazione**.

B81c.11 Consideriamo il frammento di programma seguente.

```
char lettScrlt, carDL, carrReturn;
lettScrlt='A';
carDL='\044';
carrReturn='\r';
```

Il primo dei quattro enunciati è una frase dichiarativa e stabilisce che ciascuno dei tre nomi che seguono la parola riservata `char` nel programma che segue è destinato a controllare una cella-8b; viene anche fissato l'indirizzo di tale cella, che il programmatore è ben lieto di evitare di conoscere, in quanto potrà servirsi semplicemente del corrispondente identificatore.

I tre enunciati che seguono sono frasi di assegnazione e stabiliscono quale ottetto di bits verrà inserito come valore attuale alla corrispondente cella.

Si possono usare scritture più concise: le precedenti frasi si possono sostituire con la sola seguente:

```
char lettScrlt='\100'; carDL='$'; carrReturn='\15';
```

B81c.12 In generale si possono unificare le frasi dichiarative ele frasi i inizializzazione che riguardano le stesse variabili.
Invece delle cinque frasi riguardanti variabili intere di B70d07, supposto che le frasi di assegnazione siano frasi di inizializzazione, si possono sostituire con le seguenti:

```
int step, stepNumMax = 200000000;
```

```
short matrIscritti,numIscritti=102,indScolari,numScolari=28;
```

```
long numIntntAddr=2000000000;
```

Si segnala che `short int` e `long int`, le scritture riservate usate più frequentemente, si possono sostituire con le equivalenti ridotte parole chiave r`short` e `long`.

B81 d. arrays e stringhe

B81d.01 Per affrontare moltissimi problemi si rende necessario costruire ed elaborare sequenze di dati omogenei, soprattutto sequenze di datiomogeneri.

Una sequenza di numeri naturali può servire a registrare misurazioni di grandezze ottenute in osservazioni successive come cardinali di insiemi, lunghezze, pesi, durate, velocità, quantità di denaro, temperature di un paziente, concentrazioni, percentuali, ...
Sequenze di interi consentono di fornire i numeri degli abitanti di una sequenza di città o di nazioni, le altitudini in metri di località incontrate in un dato percorso, le quantità di un prodotto fabbricato in anni successivi, i punteggi delle squadre partecipanti a un campionato, ...

Oltre alle sequenze numeriche risultano utili le sequenze di caratteri; esempi di queste sequenze sono dati dai caratteri che si incontrano in una parola di una lingua naturale, nella denominazione di una località, nel nome di una persona, nei simboli di un composto chimico, in un acronimo, ...

Il modo canonico per registrare in una memoria una sequenza di dati consiste nel collocarli in una sequenza di celle consecutive della memoria centrale.
Questo tipo di posizionamento di informazioni si può naturalmente accostare alla registrazione di dati omogenei sopra un segmento di un nastro di carta, di un nastro magnetico o di una traccia di un disco magneto-ottico.

Come si è detto in ??, ogni sequenza di celle-mm è individuata dall'indirizzo della prima e dal numero delle celle, cioè dal numero delle componenti della sequenza e un modo canonico per individuare una di queste celle si serve dell'indirizzo della cella iniziale sommato del numero di celle sulle quali si deve avanzare partendo dalla iniziale per giungere a quella richiesta.

B81d.02 Nel linguaggio C++, come sostanzialmente in tutti i linguaggi di programmazione procedurali, le sequenze in memoria sono controllate dagli **arrays**, le più semplici tra le strutture informative.
Si tratta di variabili collettive, strutture di dati per la registrazione in memoria di sequenze e altri schieramenti di dati che consentono al programmatore di accedere facilmente alle loro componenti, ossia senza preoccuparsi della loro precisa collocazione nella memoria fisica, della quale, come si è detto, si fa carico il sistema di sviluppo).
Con C++ si possono trattare arrays di una, due o più dimensioni; ora ci occupiamo solo degli arrays monodimensionali con componenti intere o costituite da bytes.

Per disporre di arrays di interi e di bytes servono dichiarazioni come le seguenti.

```
char denom[25];

short denonL,incassiN;

int incassi[50];
```

La prima frase dispone che servendosi dell'identificatore tt denom si possono trattare denominazioni di al più 25 caratteri il cui numero attuale è determinato dal valore attuale della variabile intera `denomL`.
La seconda chiede di controllare attraverso l'identificatore `incassi` sequenze di al più 50 interi il cui numero attuale sia contenuto nella variabile `incassiN`.
Inoltre resta inteso che i valori attuali dei caratteri siano elementi dell'alfabeto ASCII (visualizzabili nella gran parte delle applicazioni pratiche) e che ciascuno degli interi sia registrato in 32 bits consecutivi e quindi possano assumere tutti i valori appartenenti all'intervallo $\mathbf{[}-2^{31} : 2^{31}-1\mathbf{]}$.

B81d.03 Il programma in cui compaiono le dichiarazioni precedenti potrebbe presentare anche frasi di assegnazione come le seguenti:

```
denomL=9;
denom[0]='c; denom[1]='e'; denom[2]='l'; denom[3]='l';
denom[4]='u'; denom[5]='l'; denom[6]='a'; denom[7]='r'; denom[8]='i';
```

Queste frasi esprimono richieste operative con conseguenze simili: ciascuna di esse ha l'effetto di determinare il contenuto di un byte fornito da una **scrittura di costante**, quella che compare a destra del segno "`=`", e di assegnarlo come valore attuale a una cella-mm determinata dalla scrittura alla sua sinistra.

Va osservato che il segno "`=`" non esprime una relazione di uguaglianza e non propone una equazione, come fa quando compare in una formula matematica; esso infatti richiede una azione consistente nella assegnazione di un nuovo valore ad una variabile, cioè nella modifica del contenuto di una cella-mm.
Possiamo dire che "`=`" esprime una operazione di assegnazione e va assimilato al connettivo "`:=`" usato spesso quando si definisce una espressione matematica e talora anche per introdurre in un discorso un termine che si vuole usare specificamente.

La prima delle precedenti frasi di assegnazione comporta la determinazione della rappresentazione dell'intero 9, costituita dalla sequenza di 32 bits `00000000000000000000000000001001`, e l'inserimento di tale valore nei 4 bytes di memoria associati all'identificatore di variabile intera `denomL`.

La frase `denom[4] = 'u';` comporta invece la determinazione della rappresentazione ASCII del carattere "`u`", costituita dall'ottetto `10101110`, e l'inserimento di tale ottetto nel byte della memoria associato alla quinta delle celle per caratteri delle 25 associate all'array di caratteri `denom`, cioè alla cella che si raggiunge avanzando di 4 posizioni rispetto alla prima assegnata all'array, la quale è accessibile attraverso l'espressione `denom[0]` .

Si osserva che gli attributi ordinali che si usano discorsivamente per individuare le componenti di una sequenza (prima, seconda, terza, ...n-sima, ...) sono sfasati rispetto agli indici degli arrays nel linguaggio C++ ([0], [1], [2], ..., $n-1$, ...).

Va anche rilevato che anche l'indicazione che segue l'identificatore di un array per individuarne un componente va interpretata come azione piuttosto che come precisazione statica.
Una scrittura come `denom[4]` comporta la valutazione di un valore numerico (in questo esempio 4, ma tra le parentesi quadre si potrebbe avere anche una espressione aritmetica piuttosto elaborata), seguita da un incremento per tale valore dell'indirizzo iniziale dell'array. Questo indirizzo è misurato in bytes per `denom`, in quaterne di bytes per un array di tipo `int` e similmente per gli altri tipi di celle-m.

B81d.04 Se vogliamo disporre dei primi 10 numeri della `successione di Fibonacci (wi)` nelle prime 10 componenti dell'array che ha come identificatore `Fib` , si possono utilizzare le frasi di assegnazione che seguono.

```
FibLun=10;
Fib[0]=0; Fib[1]=1; Fib[2]=1; Fib[3]=2; Fib[4]=3; Fib[5]=5; Fib[6]=8;
Fib[7]=13; Fib[8]=21; Fib[9]=34;
```

Ciascuna di queste 11 frasi, che viene chiusa dal segno "`;`" il quale ha il ruolo di separatore di frasi, comporta l'assegnazione del valore fornito dalla scrittura decimale al secondo membro alla cella associata alla data componente di array; le successive frasi le successive coinvolgono le prime 10 celle associate all'array `Fib`.

Se dell'array `Fib` servono solo 10 componenti si può utilizzare la seguente dichiarazione più sintetica e sicura (ma più rigida)

```
int val[10] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34 } ;
```

Abbiamo visto dunque un esempio di dichiarazione e inizializzazione di un intero array di interi.
In generale una frase di questo tipo presenta nell'ordine:
la scrittura chiave che esprime il tipo di array da introdurre;
l'identificatore dell'array seguito dal numero delle sue componenti tra parentesi quadre;
il segno "`=`";
la lista dei valori da assegnare alle successive componenti separati da virgole e complessivamente delimitata da parentesi graffe.

B81**d.05** **(1) Eserc.** Predisporre nelle prime 20 componenti di un array i primi numeri primi.

(2) Eserc. Predisporre nelle 12 componenti di un array i numeri dei giorni relativi ai successivi mesi di un anno bisestile.

(3) Eserc. Porre nelle 20 componenti di un array i pesi atomici arrotondati dei 20 elementi chimici più leggeri.

B81**d.06** Le sequenze di caratteri, cioè le stringhe, rivestono grande importanza nella elaborazione dei dati.
Innanzi tutto le stringhe di caratteri visualizzabili sono necessarie per affrontare le analisi computazionali delle informazioni esprimibili nei linguaggi naturali: nomi e qualificazioni di persone, oggetti di ogni genere, prodotti, slogan, norme, ricette, tabelle di dati, sonetti, trascrizioni di registrazioni, testi di ogni genere.
Ogni testo scritto o comunque registrato può essere ridotto a una stringa (eventualmente ricorrendo ad Unicode [:d05, :d06].
Vi sono poi le informazioni che vengono espresse mediante linguaggi convenzionali e artificiali: espressioni matematiche, enunciati della logica, formule chimiche, sequenze biologiche, ricette, norme d'uso di appareecchiature, ...

Un vasto campo applicativo delle stringhe riguarda la elaborazione automatica dei testi di ogni genere e in particolare dei testi che sono utilizzati nei prodotti software di più evidente utilità: i testi sorgente dei linguaggi di programmazione e di gestione delle basi dati, i testi in grado di governare il tracciamento di figure i testi dei linguaggi per la tipografia, per la grafica e per la comunicazione; alcuni esempi: TEX `(we)`, `HTML (we)`, `XML (we)` ed `SVG (we)`.

B81**d.07** Riconosciuta l'importanza delle stringhe, i linguaggi C e C++ hanno reso disponibili vari strumenti per il trattamento delle stringhe.

Le prime componenti di C che consideriamo sono gli **string literals**, le scritture che consentono di definire concisamente delle stringhe costanti. Queste scritture sono costituite da sequenze di caratteri ASCII racchiuse tra doppi apici come

```
"stringa"  ,  "Avvertimento"  ,  "testo costituito da 32 caratteri"
```

Con string literals si possono trattare anche stringhe molto lunghe che conviene scrivere utilizzando più linee del file che fa da supporto fisico del testo sorgente: basta per questo far comparire come ultimo carattere di una linea di testo il carattere backslash "\" .

```
"Questa frase piuttosto, lunga e verbosa, come lorsignori possono \
```

```
constatare direttamente, viene scritta su tre linee del file .txt che fa \
da supporto per il testo davanti ai vostri occhi."
```

In uno string literal possono comparire anche caratteri ASCII non visualizzabili. In particolare si possono avere caratteri con effetti tipografici come new line, carriage return ed horizontal tab. Per esempio uno string literal contenente occorrenze del carattere new line rappresentato da "\n" se inviato a una stampante comporta la emissione di più linee.

String literals nei quali compaiono molti caratteri di controllo possono avere effetti di stampa o di emissione su video piuttosto elaborati. Con questi literals vengono generati i disegni che costituiscono la cosiddetta `ASCII art (we)`.

Ogni string literal viene collocato in una sequenza di bytes occupati dai successivi caratteri tra i doppi apici seguiti da un ottetto contenente il carattere null, il primo dell'alfabeto ASCII costituito da otto bit uguali a 0. Le stringhe che presentano un null finale qualche vantaggio pratico e sono chiamate **null teminated strings**.

B81**d.08** Osserviamo che se avessimo voluto trattare il nome del matematico Izrail Gelfand nella sua versione cirillica o nella yiddish non sarebbe sufficiente l'alfabeto ASCII e si dovrebbe fare ricorso al più generale sistema `Unicode (wi)`.
Similmente se avessimo voluto trattare i primi 10 `numeri di Mersenne (wi)` non sarebbero stati sufficienti 10 celle di memoria dedicate a interi dell'intervallo $[2^{31} : 2^{31} - 1]$, in quanto solo i primi 8 numeri sono trattabili con queste celle (l'ottavo è $M_{31} = 2^{31} - 1$, mentre il nono, $M_{61} =$ `2305843009213693951`, e il decimo, $M_{89} =$ `618970019642690137449562111`, superano la capacità di tali celle.

Se si vogliono trattare numeri interi molto elevati, come vedremo, si devono adottare modalità di codifica più complesse, oppure rinunciare alla precisione ed accontentarsi di valutazioni approssimate, ottenibili i cosiddetti numeri real che vedremo più avanti, le quali non sono in grado di soddisfare certe esigenze e di risolvere certi tipi di problemi; ad esempio sono del tutto inutilizzabili per trattare problemi di natura crittografica.

Ribadiamo che i limiti finiti degli strumenti effettivi, ovviamente, devono essere tenuti ben presenti nella pratica computazionale.
Tali limitazioni nello sviluppo di considerazioni generali, viceversa, possono essere considerate quasi un intralcio alle formulazioni rapidamente comprensibili di molti enunciati e di molte argomentazioni. In effetti gli atteggiamenti di chi porta avanti studi matematici specialistici e di chi si occupa di calcoli specifici possono essere piuttosto diversi e in certi aspetti contrastanti.

B81 f. entrata e uscita

B81f.01

```
int x;
const pigr 3.14 ;
cout << "Digita il valore del raggio:  "; // chiede il raggio di una sfera
cin >> x; // ottiene il valore chiesto dalla tastiera della console
// calcola volume e superficie della sfera e li emette
cout << "volume :  " << x; << "superficie della sfera :  "
```

La nozione di stream

L'esposizione in https://www.mi.imati.cnr.it/alberto/ e https://arm.mi.imati.cnr.it/Matexp/matexp_main.php