1

Capitolo B71
linguaggio di programmazione c++ [2]

Contenuti delle sezioni

15 pagine

B710.01 Questo capitolo, il secondo sulla introduzione al linguaggio c++, inizia [:a] introducendo i numeri di solito chiamati "numeri reali", ma che in realtà costituiscono un sottoinsieme finito di $\mathbb{Q}$ e che qui chiamiamo **numeri real-fp**.

Questi consentono di servirsi delle espressioni in grado di controllare nel modo più diretto le costruzioni computazionali della geometria, dell'analisi infinitesimale, della fisica e di tutte le loro applicazioni quantitative.
Queste costruzioni per la maggior parte portano a risultati che al livello idealizzato della matematica sono numeri reali, mentre al livello della effettività operativa si devono servire dei numeri real-fp.
I problemi che conducono a risultati per i quali bastano i numeri interi sono relativamente pochi e in genere riguardano problemi con finalità strettamente organizzative (ad esempoio quelle dalla ricerca operativa) che si trovano nei più vasti problemi applicativi con ruoli di sottoproblemi circoscritti.

Successivamente, in :b e in :c, si affronta un importante ampliamento della la gamma dei dati controllabili con il linguaggio C++ introducendo gli arrays di una e di più dimensioni, entità che aprono rilevanti possibilità per il controllo delle grandi quantità di dati e per l'automatizzazione delle elaborazioni dei processi entro ambienti modellabili con la geometria cartesiana.

La sezione :d riguarda l'introduzione dei sottoprogrammi e la organizzazione modulare dei programmi, un altro fondamentale ampliamento della portata della programmazione che porta alla scalabilità quasi illimitata delle elaborazioni con automatismi.

Il primo genere di sottoprogrammi che si esamina con una certa ampiezza [:e] è quello che si rivolge alle manovre per l'entrata e l'uscita dei dati.

L'ultima parte consiste nella discussione di vari piccoli programmi che risolvono problemi circoscritti che si possono incontrare in vari contesti e quindi si possono considerare di rilevante importanza propedeutica.

B71 a. numeri ed espressioni reali

B71a.01 Un importante tipo di dati è quello che fornisce la possibilità di servirsi di una gamma, necessariamente finita ma estesa, di numeri reali che qui chiameremo tipo dei **numeri real-fp**.

In effetti questo insieme di configurazioni binarie rappresenta fedelmente un certo insieme di numeri razionali che stiamo per individuare con precisione, ma attraverso considerazioni assai minuziose e dettagliate.

Occorre anche dire che spesso questi numeri vengono chiamati sbrigativamente "numeri reali" e vengono proposti senza vere definizioni, ma basandosi sul fatto che funzionano piuttosto bene per approssimare i numeri reali della matematica che servono per risolvere una significativa gamma di problemi di interesse pratico.
Questo modo di procedere, giustificato per gran parte delle attività computazionali mediamente impegnative, lascia margini di attendibilità che nei progetti di elevata responsabilità devono essere esaminati con studi specificie e risolti con tecniche che particolari.

B71a.02 Per introdurre i numeri reali-fp conviene partire dai literals che consentono di esprimerli nel testo del programma e dalle configurazioni di bits che costituiscono le loro conseguenti implementazioni nelle celle-mm e che li rendono effettivamente manipolabili.

Un numero real-fp positivo non troppo grande, né troppo piccolo, viene espresso con la notazione che presenta la sua parte intera, il separatore "." e la sua parte decimale.
Esempi `15.085 , 0.000253 . 250000000`
Questa notazione permette di trattare agevolmente numeri il cui ordine di grandezza va dal miliardesimo al miliardo.
In molti calcoli scientifici servono anche numeri ben inferiori al miliardesimo e ben superiori al miliardo.
Per questi si rende necassaria la notazione esponenziale che alla parte intera, al punto e alla parte decimale fa seguire un fattore costituito da una potenza positiva o negativa di 10.

La massa dell'elettrone si conviene che sia $9,109\,383\,701\,5 \cdot 10^{-31}$ kg.
La costante di Avogadro, il numero delle particalle costituenti una mole di sostanza, è definita come $N_A := 6,022\,140\,76 \cdot 10^{23}$ mol^{-1}

B71 **b. arrays monodimensionali**

B71**b.01**

B71**b.02**

B71 c. arrays multidimensionali e geometria cartesiana discreta

B71c.01 Per rappresentare una grande varietà di situazioni servono gli **arrays a due indici**, schieramenti strutture di dati che consentono di implementare le matrici, entità matematiche con una grande quantità e varietà di applicazioni.

Nel linguaggio C gli arrays a due indici si possono considerare arrays a un indice le cui componenti sono a loro volta degli arrays a un indice, allineamenti di dati tutti costituiti da uno stesso numero di componenti elementari.

Volendo disporre nella sottomatrice 4×4 di una matrice 10×10 dei primi coefficienti binomiali simmetrici [??] si possono utilizzare le seguenti frasi (la prima dichiarativa le successive di assegnazione)

```
int cbs[10][10];
cbs[0][0]=1; cbs[0][1]=1; cbs[0][2]=1; cbs[0][3]=1;
cbs[1][0]=1; cbs[1][1]=2; cbs[1][2]=3; cbs[1][3]=4;
cbs[2][0]=1; cbs[2][1]=3; cbs[2][2]=6; cbs[2][3]=10;
cbs[3][0]=1; cbs[3][1]=4; cbs[3][2]=10; cbs[3][3]=20;
```

Se in un programma è sufficiente disporre di una matrice 4×4, basta la seguente frase di inizializzazione

```
int cbs[4][4] = {
                    {1, 1, 1, 1}
                    {1, 2, 3, 4}
                    {1, 3, 6, 10}
                    {1, 4, 10, 20}
                }
```

A questo arrays si dedicano 16 celle-mm per interi normali, celle-16B.
In questa sequenza di celle si devono distinguere 4 sottosequenze, ciascuna costituita da 4 celle: la prima sottosequenza è dedicata alla prima riga della matrice, accessibile per intero scrivendo `cbs[0]`, la seconda alla seconda riga accessibile mediante la scrittura `cbs[1]` e così via.

B71c.02 Talora servono arrays con 3 o più indici: con arrays a 3 indici si possono trattare grandezze fisiche attribuibili a una griglia spaziale, ad esempio le temperature in un certo numero di punti all'interno di una caldaia a forma di cuboide. L'organizzazione in memoria di tali complessi di dati viene effettuata generalizzando il procedimento precedente.
Un array a tre dimensioni potrebbe essere introdotto con una dichiarazione come la

```
int valspaz[5][3][7];
```

la quale comporta la predisposizione di un array con 5 componenti ciascuna delle quali è un array bidimensionale di profilo 3×7.

B71 d. sottoprogrammi e richiami delle functions

B71d.01 Per affrontare problemi di interesse pratico si devono scrivere programmi di molte migliaia e anche di molti milioni di istruzioni.

In effetti lo sviluppo delle apparecchiature elettroniche programmabili iniziato a metà del secolo scorso ha portato alla messa a punto di procedure molto elaborate e la tecnologia della programmazione costituisce da decenni uno dei fattori industriali chiave per lo sviluppo dell'economia.

Le attività concernenti la produzione di programmi sono ormai sviluppate con criteri industriali e in esse si rende necessaria una sistematica suddivisione dei compiti; questa suddivisione riguarda sia le azioni svolte dai programmi, sia le attività delle persone che i programmi progettano, redigono, collaudano, fanno evolvere e analizzano criticamente.

Per la suddivisione delle azioni controllate dai programmi, quali che siano i linguaggi di programmazione utilizzati, assume grande importanza la organizzazione dei sottoprogrammi.

B71d.02 Un sottoprogramma si può definire come una porzione di un programma chiaramente delimitata, sia al livello del testo che costituisce la sua formalizzazione, che al livello semantico del suo significato operativo, ossia del suo compito di effettuare un servizio specifico in risposta di ciascuna richiesta che gli viene rivolta.

Ciascuna richiesta a un sottoprogramma ***S***, in linea di massima, contiene specificazioni sulle particolari prestazioni che gli vengono richieste.

Le prestazioni richieste a un sottoprogramma devono essere adeguate alla portata operativa consentita al sottoprogramma stesso.
Tuttavia i sottoprogrammi in generale, in accordo con la portata complessiva della programmazione e con l'efficienza e la versatilità delle risorse oggi disponibili, sono in grado di effettuare una vastissima varietà di servizi.
Ed in effetti tutti i programmi che risolvono problemi applicativi di interesse concreto demandano la gran parte delle loro prestazioni a un numero rilevante di sottoprogrammi.
Molti programmi mediamente elaborati negli obiettivi e nelle prestazioni richiedono l'intervento di molte migliaia di sottoprogrammi con compiti e caratteristiche anche molto differenziate.

B71d.03 A questo punto facciamo riferimento ad alcuni esempi che riguardano sottoprogrammi dei tipi che incontreremo più spesso nel seguito.

Ad un programma di calcolo tecnico-scientifico sono necessari vari sottoprogrammi incaricati di effettuare calcoli specifici:
calcoli di funzioni matematiche che avremo modo di incontrare più volte (radici quadrate, funzioni esponenziali e trigonometriche, altre funzioni speciali, trasformazioni di coordinate, conversioni, ...);
soluzioni di sistemi di equazioni lineari e loro varianti; determinazione di zeri di polinomi e di altre funzioni di variabile reale; elaborazioni di figure geometriche, di processi meccanici, di evoluzioni finanaziarie, di analisi statistiche, ...;
manovre su strutture combinatorie e in particolare su grafi variamente arricchiti (per esempio grafi in grado di rappresentare reti di trasporto, evoluzioni di automi, schemi organizzativi, ... ;
generazioni di grafici e di animazioni che consentono di presentare i risultati di elaborazioni tendenzialmente complessi e numerosi attraverso immagini la cui comprensione è aiutata dalla intuizione visiva; in particolare possono esser d'aiuto grafici che forniscono dati di sintesi statistiche riguardanti

i parametri di sintesi relativi a popolazioni di migliaia o milioni di esemplari; possono anche essere molto efficaci le animazioni riguardanti le evoluzioni di processi materiali o di comportamenti umani.

B71d.04 Alle operazioni di ingresso e uscita vengono dedicati tanti tipi di sottoprogrammi.
Molti di questi si incaricano del controllo dei dispositivi hardware delle molteplici apparecchiature che si possono connettere a una macchina programmabile.
Oggi si possono considerare periferiche tradizionali i terminali video con tastiera e mouse, le stampanti, i plotters, i dischi, i nastri, le memorie flash.
A queste in tempi più recenti si sono aggiunti le apparecchiature per l'immissione e l'emissione dei suoni, le macchine fotografiche, le videocamere, monitors televisivi e i collegamenti con reti della telefonia e reti di computers.
Tutte queste apparecchiature richiedono routines agenti sull'hardware che evidentemente dipendono da una varietà di caratteristiche tecniche, richiedono programmatori con competenze deinite e approfondite e possono anche richiedere linguaggi specifici.

Venendo a prestazioni più vicine alle applicazioni e formulabili con linguaggi come C++, per le operazioni di ingresso dei dati ricordiamo: lettura di stringhe e loro codifica in valori interni, tipicamente lettura di scritture che seguono notazioni standard di numeri interi o razionali; lettura di denominazioni leggibili e loro trasformazioni in codifiche convenzionali che rendono più agevole la determinazione di loro caratteristiche (ad esempio le codifiche con una lettera e 3 cifre dei comuni italiani).
Ancora più variegate sono le prestazioni di scrittura: operazioni di decodifica in chiari speculari delle accennate operazioni di codifica; presentazione di grafici che raffigurano dati numerici (istogrammi, areogrammi, ...); arricchimento dei risultati con dati di sintesi (medie, varianze, totali parziali, ...); generazione di animazioni.

Assimilabili ai sottoprogrammi di entrata e uscita sono i sottoprogrammi che si incaricano di operazioni di inserimento e di reperimento di dati entro strutture informative e archivi.
Molti altri sottoprogrammi si possono incaricare di conversioni e di transcodifiche relative a formati e a strutture di dati definiti per rendere calcolabili determinati modelli e per trattare più agevolmente particolari manovre; in particolare vi sono programmi che organizzano le "visite" di strutture informative per analizzarle o individuarne elementi di rilievo (minimi, massimi, estremali, ottimali).
Per concludere, non possiamo non ricordare i sottoprogrammi che si occupano della gestione di situazioni erronee o anomale, operazioni che nei più recenti linguaggi di programmazione hanno ottenuto specifici riconoscimenti anche al livello della sintassi.

B71d.05 Nel linguggio c++ i sottoprogrammi assumono la forma di quelle che chiameremo **functions**, entità con alcune caratteristiche delle funzioni della matematica, ma per le quali useremo il termine inglese per sottolinearne le differenze.

Una function è una sezione sintatticamente ben definita del testo sorgente di un programma che può essere richiamata per eseguire manovre specifiche che è opportuno siano ben definite. La forma di una function prevede una intestazione e un corpo e si può schematizzare come segue.

tipo-della-function *identificatore-della-function* (*lista-degli-argomenti*)
{
corpo-della-function
}

Ogni esecuzione di una function può produrre un risultato primario che risulta associato al suo richiamo ed è disponibile nell'espressione che contiene il richiamo stesso.

Nelle pagine che seguono incontreremo prevalentemente functions che producono un risultato primario intero. Sono comunque importanti le functions che producono un risultato primario dei tipi `double` e `char` e quelle che producono un indirizzo di dati singoli o multipli dei vari tipi.
A una fuction si attribuisce il tipo del dato che produce e che va dichiarato esplicitamente nel programma.
Occorre tuttavia segnalare che alcune functions non producono un risultato esplicito e ad esse va attribuite lo specifico tipo corrispondente alla parola riservata `void` .

B71d.06 Gli identificatori delle functions seguono le stesse regole lessicali degli identificatori delle variabili e come per questi si pone il problema della non omonimia.

È opportuno che l'identificatore di ogni function sia scelto in modo da evidenziare il suo compito; questa scelta, in genere piuttosto delicata, va fatta cercando di avere chiari i suoi possibili utilizzi e quali altre functions saranno utilizzate in collegamento o in alternativa a essa.

La lista degli argomenti di una function può contenere uno o più argomenti o anche nessun argomento. Di ogni argomento si deve esplicitare il tipo e un identificatore può essere accompagnato da un modificatore.

Il corpo di una function ha forma simile a un blocco di istruzioni. In genere inizia con dichiarazioni di variabili locali la cui visibilità è limitata al solo corpo della function.

Successivamente può presentare una qualsiasi composizione di sequenze di comandi e di blocchi per selezioni e per iterazioni. È opportuno che queste composizioni siano ben strutturate.

Nel corpo di una function deve comparire almeno una istruzione `return` accompagnata da un argomento costituito da un'espressione che fornisce un risultato avente lo stesso tipo della function.
L'esecuzione di un comando `return` fornisce come risultato primario di ogni richiamo della function il valore attualmente calcolato per l'espressione.

B71d.07 Ogni richiamo alla function è costituito dal suo identificatore seguito dalla cosiddetta **sequenza degli argomenti attuali**, espressioni costituite da operandi valutabili all'atto dell'esecuzione del richiamo le quali dovranno essere in grado di fornire un valore per ciascuno degli argomenti formali della intestazione della function richiamata.

Una function per essere utilizzabile in un modulo di programma deve essere dichiarata nel testo di tale modulo.
Una dichiarazione di una function può essere una semplificazione della sua intestazione nella quale ogni argomento viene sostituito dalla semplice dichiarazione del suo tipo.
Questo enunciato semplifica il lavoro al traduttore del testo sorgente del programma nel suo equivalente utilizzabile per l'esecuzione (dopo la manovra di linkaggio dei vari moduli che possono costituire un programma che vedremo più oltre), in quanto annuncia le caratteristiche essenziali, cioè i tipi, dei suoi argomenti e del suo risultato primario.
Alla dichiarazione devono adeguarsi l'intestazione della function e tutti gli enunciati nei quali viene richiamta.

In una dichiarazione di function la specificazione di un argomento può presentare, oltre al suo tipo, un identificatore fittizio che conviene scegliere in modo di fornire a chi legge il testo del modulo, un'indicazione del ruolo dell'argomento stesso, ovvero del suo significato operativo.
In tal modo una dichiarazione di function riesce a documentare in forma concisa il suo significato.

B71d.08 Anche il modulo principale di ogni programma viene strutturato come una function. Tuttavia l'identificatore di tale function è prefissato, deve essere la parola riservata `main`.

Anche il modulo `main` può avere un tipo, ma spesso si riduce a `void`.

La forma dell'intestazione di un modulo principale è la seguente:

tipo `main(int argc, char **argv)`

Si prevede che la richiesta dell'esecuzione di un programma sia ottenuta digitando il nome del file contenente il cosiddetto **programma oggetto**, ricavato dal testo sorgente dal linker, eventualmente seguito da un determinato numero di stringhe che sono in grado di indirizzare la sua esecuzione.
Il numero di tali stringhe è fornito dalla variabile identificata da `argc` (sta per "argument count") ed esse sono reperibili in sequenze di bytes le cui `argc` posizioni iniziali sono ottenibili dall'array il cui identificatore è `argv` (sta per "argument values").
Questo meccanismo e il ruolo di "`**`" verrà precisato più oltre.

B71d.09 Riprendiamo gli effetti che può avere l'esecuzione di un richiamo di function.
Per questo occorre distinguere tra gli argomenti della function quali vengono "richiamati per valore"' e quali sono "richiamati per indirizzo"'.

All'inizio dell'esecuzione di un richiamo di function ciascuno degli argomenti fornisce un valore che viene fornito alla function.
Ciascuno degli argomenti costituiti da espressioni da valutare o da variabili fornisce un valore che viene riprodotto nella corrispondente variabile fittizia.
A sua volta ciascuno degli argomenti che individuano un indirizzo porta alla riproduzione di tale indirizzo in una variabile interna alla function.
La riproduzione di un argomento (richiamato per valore) costituito da una variabile assicura che la function non sia in grado di modificare il valore di questa variabile nel modulo richiamante. Se questo accadesse si avrebbe una modifica nel modulo richiamante non evidenziata dal suo testo e quindi poco controllabile e con il rischio di un fraintendimento.

La riproduzione degli indirizzi lascia invece aperta la possibilità di modificare i contenuti delle celle di memoria associate agli indirizzi ed accessibili sia al modulo richiamante che alla function.
Queste celle devono essere utilizzate come indirizzi e quindi le modifiche che la function può effettuare sono maggiormente evidenti e controllabili dai programmatori del modulo richiamante e della function delle modifiche che si potrebbero effettuare all'interno di una function che potesse agire direttamente sulle celle nelle quali sono registrati i valori degli argomenti.

Tutta questa organizzazione è indubbiamente elaborata, ma ha lo scopo di rendere meno probabili le modifiche di dati del modulo chiamante da parte di istruzioni formulate nel testo sorgente della function.

La function dopo la riproduzione degli argomenti, procede a eseguire le manovre formulate nel blocco costituente il corpo della function. Queste manovre potrebbero essere molto complesse, anche perché molte functions che prestano servizi complessi in genere richiamano altre functions e questa organizzazione di functions da collocare a diversi livelli può essere assai articolata.
Le manovre implicate da un richiamo di function forniscono il servizio che costituisce la ragion d'essere della function stessa.
Va sottolineato che una function può servirsi delle componenti di più arrays e può modificarle, ma solo attraverso gli indirizzi iniziali di questi schieramenti di dati.

B71d.10 Consideriamo il programmatore P_M di un modulo **M** che richiama una function **F** curata da un secondo programmatore P_F (caso simile a quello in cui si tratta dello stesso programmatore $P_M = P_F$ il quale prima ha scritto **G** e solo dopo parecchio tempo deve curarsi di **M**).

Per servirsi correttamente della **F** in **M**, è necessario che gli effetti sui suoi argomenti a P_M risultino definiti con precisione e completezza.
Viceversa non è necessario che P_M abbia presenti i dettagli delle operazioni previste in **F**.
Anzi, in linea di massima è opportuno che questi effetti siano documentati solo nel testo sorgente della function o in un documento a essa associato e non compaiano nel testo del modulo chiamante in quanto potrebbero complicarne la comprensione.
Infatti ogni function è tanto più utile quanto più agevolmente il programmatore responsabile di qualche modulo che la richiama riesca a tenere sotto controllo i suoi richiami.
Questo programmatore P_M deve potersi documentare in modo completo e preciso, ma limitatamente agli effetti per l'esterno della function e deve riuscire ad evitare di addentrarsi nei dettagli delle operazioni che vengono effettuate nel corso delle esecuzioni della function.

Quindi si favorisce una divisione dei compiti tra il programmatore della function e il programmatore delle sue utilizzazioni: per un buon utilizzo delle risorse umane in una attività impegnativa, costosa e carica di responsabilità come la programmazione è bene che i dettagli delle operazioni effettuate dalla function possano rimanere nascosti ai programmatori che devono solo utilizzarle attraverso i loro richiami.

Questa ripartizione dei compiti si realizza con il cosiddetto **incapsulamento** delle manovre di una function.
Va ribadita l'importanza della documentazione verso l'esterno delle prestazione delle functions: questa deve essere precisa, completa e rapidamente leggibile, soprattutto quando di una function complessa si vogliano utilizzare solo prestazioni circoscritte.
Tenuto conto dell'importanza in quanto prodotti industriali delle functions in un linguaggio di programmazione di largo uso, dovrebbe essere chiaro che nella pianificazione della messa a punto di un programma si devono dedicare molte attenzioni alla definizione e alla scelta di ciascuna una cosiddetta library delle functions utilizzabili, alla documentazione delle loro prestazioni e alle prove sistematiche che possono garantire la loro correttezza e la loro adeguatezza.

B71d.11 Aggiungiamo alcune prestazioni perticolari dei richiami di function.

Si possono usare anche frasi esecutive della forma

espressione **;**

L'effetto di questa frase è la valutazione dell'espressione, processo che può comportare i cosiddetti **effetti collaterali** (*side effects*), cioè modifiche di variabili che possono avere un ruolo importante nelle elaborazioni successive.

Quando un richiamo di function viene seguito dal solo "**;**" il risultato della valutazione della function, se effettivamente costruito, resta inutilizzato; avranno seguito serviti solo gli effetti collaterali che nel modulo chiamante non compaiono e possono restare nascosti a chi esamina il programma.

Gli effetti collaterali dei richiami di function possono essere di vari generi.
Possono essere eseguite varie manovre sopra unità periferiche come riavvolgimenti di nastri, aperture e chiusure di files, letture di rilevanti gruppi di dati e scritture di complessi di risultati.
Possono essere eseguite manovre rilevanti su variabili nella memoria centrale o disponibili sullo spazio chimato heap.
Possono essere inviati messaggi digitali o segnali d'altro genere con conseguenze sull'esterno che possono portare a reazioni sull'andamento della prosecuzione della esecuzione; in particolare si possono avere interazioni con il Web.

Inoltre si possono organizzare functions senza risultati di ritorno, le cosiddette **non-value returning functions**.

Per questi richiami occorre raccomandare che l'effetto del richiamo sia ben chiarito dal suo programmatore agli operatori interessati ai risultati che possono essere influenzati e ad altri programmatori che fossero incaricati di aggiornare o comunque modificare il modulo contenente queti richiami.

B71 e. organizzazione modulare dei programmmi

B71e.01 Occorre segnalare che per sviluppare programmi applicativi incisivi in genere servono ampie librerie di functions e che è di grande importanza la facilità del loro riutilizzo per applicazioni diverse da quelle che hanno condotto alla loro prima stesura.

La possibilità di disporre di ampie librerie come accade per i sistemi di sviluppo dei linguaggi che contano su una diffusione ampia e consolidata nel tempo costituisce un elemento di grande importanza per le attività di sviluppo del software e in particolare per la scelta di un linguaggio per ognoi progetto impegnativo.

La disponibilità di librerie di sottoprogrammi versatili e affidabili si può considerare come un importante arricchimento delle prestazioni del linguaggio stesso.
Per un buon uso di queste librerie di programmi è opportuno disporre anche di strumenti che consentano di reperire efficacemente le functions che possono servire. Attualmente sono disponibili vari sistemi di sviluppo per i linguaggi di programmazione più diffusi che posseggono strumenti efficaci e versatili per la gestione delle librerie.
Per lo sviluppo dei programmi da alcuni anni si possono consultare efficacemente anche vari siti Web, in particolare siti attraverso i quali si possono ottenere suggerimenti da intere comunità di programmatori.
Queste considerazioni vogliono solo inquadrare il problema della gestione dei sottoprogrammi e le problematiche della programmazione modulare. Chi intende occuparsi professionalmente di questi problemi dovrà approfondirli adeguatamente.

Qui nel seguito concretamente ci interesseranno invece functions di portata limitata, con finalità ben definite che consentano di esprimere in modo preciso e verificabile una gamma di algoritmi che sia significativa per le nozioni matematiche e computazionali che esamineremo.
Va comunque segnalata la vicinanza concettuale tra la disponibilità dei risultati della matematica attraverso terminologie e notazioni ampiamente condivisibili e le linee generali seguite per la organizzazione delle librerie di sottoprogrammi riutilizzabili per lo sviluppo di prodotti software.
Questa vicinanza si evidenzia soprattutto quando si considerino i sottoprogrammi per la soluzione di problemi computazionali come implementazioni di risultati matematici.

B71e.02 Gli odierni ambienti per lo sviluppo dei programmi scritti in un linguaggio di programmazione di largo uso mettono a disposizione ampie librerie di sottoprogrammi predisposti per risolvere problemi che si presume si debbano affrontare spesso.
Le librerie di sottoprogrammi possono anche essere messe a punto dai programmatori che affrontano problemi specifici operando singolarmente o in gruppi di lavoro, oppure essere acquisite da fornitori specializzati che rendono disponibili prodotti software di largo interesse o su misura, oppure possono essere reperite nei siti del Web finalizzati alla messa a disposizione di software libero.

Nel linguaggio C++ vengono rese disponibili effettivamente molte librerie di functions mediante i comandi `#include` .
Questi sono enunciati di un preciso tipo detto tipo dei **comandi per il preprocessore**; sono frasi ccaratterizzate dal fatto di iniziaro con il carattere `#`, che ciascuno di assi occupa una linea del sorgente del modulo nel quale compare, prima dell'enunciato che inizia con la parola riservata `main`.

Tra le librerie di base per C++ segnaliamo la `iostream`, la tt conio e la `stdio` riguardanti prestazioni di ingresso e uscita e la `math` che raccoglie functions dedicate a operazioni matematiche.

B71e.03 Presentiamo alcune routines per la manipolazione di stringhe.

Preliminarmente va detto che in C/C++ vengono trattate facilmente le stringhe ASCII organizzate in arrays del tipo `char` monodimensionali che fanno seguire i successivi da un carattere `null` che segnala la conclusione della stringa stessa. Una stringa di n caratteri deve avere a disposizione un array di almeno $n + 1$ bytes.

`strcat(str1,str2)` Concatena, ossia giustappone, due stringhe.

`strcmp(str1,str2)` Confronta le due stringhe argomento e fornisce 1 sse coincidono, 0 in caso contrario.

`strcmpi(str1,str2)` Confronta le due stringhe argomento e fornisce 1 sse coincidono oppure presentano differenze maiuscola/minuscola, 0 in caso contrario.

`strcpy(str1,str2)` Riproduce la stringa assegnata a `str2` nell'array `str1`.

`strstr(str1,str2)` Scandisce la stringa in `str1` alla ricerca della prima occorrenza come sua sottostringa della stringa in `str2`.

`strlen(str1)` Fornisce l'intero esprimente la lunghezza della stringa argomento.

`strupr(str1,str2)` Pone in `str1` la stringa fornita da `str2` dopo aver modificato ogni eventuale carattere minuscolo nel corrispondente minuscolo.

`sprintf()` Costruisce una stringa a partire da dati in numero variabile, cioè che in diverse frasi di richiamo possono presentare argomenti in diversi numeri.

B71e.04 Tra le functions, e in particolare tra le precedenti, occorre distinguere tra quelle che non presentano e quelle che presentano i cosiddetti **effetti collaterali**. L'effetto di un richiamo di una function del primo tipo consiste solo nella fornitura di un valore del tipo assegnato alla function nella sua dichiarazione. Viceversa una function presenta effetti collaterali se un suo richiamo comporta modifiche alle variabili e agli arrays che del richiamo sono gli argomenti.

Tra le functions del paragrafo precedente non presentano effetti collaterali `strcmp`, `strcmpi`, `strstr` e `strlen`.
Ne comportano invece `strcat`, `strcpy`, `strupr` e `sprintf`.

B71 f. sottoprogrammi per l'entrata e l'uscita

B71f.01 Le operazioni di entrata e uscita si possono programmare servendosi di diverse modalità, ovvero utilizzando gruppi di sottoprogrammi facenti parte di diverse librerie. Tra queste si deve distinguere, innanzi tutto, tra quelle introdotte con il linguaggio C (tendenzialmente più elementari) e quelle adottate con il linguaggio C++ (che in genere sono di uso più impegnativo, ma sono più complete e danno maggiori garanzie di sicurezza).

Per usare questi sottoprogrammi si deve provvedere all'inserimento dei comandi iniziali per la predisposizione degli header della forma

`#include <`*specifica di header*`>`

L'header per i tradizionali sottoprogrammi di I/O del linguaggio C ha come specifica `stdio.h`.
Nell'ambito dei sistemi di sviluppo per C++ sono disponibili i seguenti headers:

`iostream.h` riguarda sottoprogrammi per l'entrata da e l'uscita su i cosiddetti streams, periferiche o files che gestiscono sequenze di bytes.

`iomanip.h` fornisce prestazioni di manipolazione dei dati;

`fstream.h` riguarda operazioni per l'emissione su e l'immissione da files su memorie di massa.

B71f.02 Le più semplici operazioni riguardano l'immissione e l'emissione di singoli caratteri attraverso le functions `getchar` e `putchar`. Vediamo le schema di un semplice programma che si serve di tali functions.

```
#include <stdio.h>
main();
char c,d;
c = getchar();    d = getchar();    // immissione dei due caratteri
if(('a'<=c && c<='Z') || ('a'<=c && c<='z') {// in c lettera
    if('0'<=d && d<='9') {   // in d cifra:  caratteri accettati.
    putchar(c); putchar(' '); putchar('e'); putchar(d);
        putchar(' '); putchar('o'); putchar('k'); putchar(EOF);
        return(1);
        }
    }
// caratteri rifiutati.
putchar('i'); putchar('i'); putchar('n'); putchar('p'); putchar('u');
putchar('t'); putchar(' '); putchar('n'); putchar('o'); putchar(EOF);
return(0);
}
```

Da questo programma si intuisce come mediante la lettura dei singoli caratteri dei dati si possono controllare tutti i loro dettagli. Si vede anche come si possano precisare le emissioni, ma risulta evidente anche la minuziosità di questo modo di organizzare letture e scritture e di conseguenza l'opportunità di disporre di strumenti che consentano controlli più sintetici.

B71f.03

B71f.04

B71f.05

B71f.06

B71 g. esempi di programmi e di elaborazioni

B71g.01 Iterazioni annidate e visita di matrici e di griglie tridimensionali

B71g.02

B71g.03

B71g.04

B71g.05

B71g.06 Conversioni di rappresentazioni Dati numerici, testi

B71g.07 Funzioni date da espressioni aritmetiche a tratti

B71g.08

B71 h. programmi su interi e stringhe [1]

B71h.01 Il seguente programma calcola un coefficiente binomiale.

B71h.02 Programma per la fattorizzazione di un intero positivo.

B71h.03 Calcolo del minimo comune multiplo e del massimo comun denominatore.

B71h.04 Elenco di numeri primi.

B71h.05 Precisazione della posizione settimanale di un giorno

B71h.06 Taglio sillabico di parola italiana.

B71h.07 Conversioni di una data.

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