1

Capitolo J11
c++, primo approccio

Contenuti delle sezioni

24 pagine

J11 0.01 Questo è il primo dei capitoli dedicati al linguaggio di programmazione procedurale C++; seguono J11, BJ2, B83, J14 e J16.

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

Innanzi tutto la sua valenza 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 non sono pochi i programmi in C++ presenti nella *esposizione*, non solo nei cinque capitoli citati, ma anche in stretta connessione con la trattazione di nozioni strettamente matematiche.

La seconda motivazione riguarda il fatto che C++ si può vedere come corrispondente effettivamente sperimentabile, del linguaggio delle macchine sequenziali programmabile, ossia del genere di strumenti che nei capitoli B17, B18, e B19 introducono la computabilità, cioè la nozione che nella *esposizione* ha un ruolo primario per la illustrazione delle finalità e di alcune 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 A05 e si ritiene in grado di sostenere molte altre procedure prospettabili per questi scopi.

J11 0.02 Conviene ricordare che C++ discende dal linguaggio C ampliando notevolmente la sua portata. Non si tratta di una pura estensione in quanto alcune, poche, caratteristiche formali di C non sono accettate da C++.

Occorre tenere presente che C++ continua ad evolversi nel tempo in quanto intende sostenere generi di programmazione che si vengono ad imporre in relazione alle continue innovazioni nell'hardware e nello scenario applicativo; il sostegno alla programmazione orientata agli oggetti è stata il motivo della sua prima definizione da pare di Bjorn Stroustrup.

In particolare C++ si avvale di librerie standard molto più ampie di quelle disponibili a chi programma con il linguaggio C e che continuano ad errichirsi il linguae dall'essere in progressiva evoluzione.

Nel seguito introdurremo solo una parte delle prestazioni del linguaggio C++ e per mettere in rilievo questa limitazione in seguito talora parleremo della presentazione di un linguaggio che chiamiamo **c++**, trigramma che vuole denotare un sottoinsieme ridotto del linguaggio C++.

Questo linguaggio ufficioso c++ è quasi completamente contenuto nel linguaggio C; la sua portata è contenuta quasi interamente nella portata di C, ma comprende alcune prestazioni di C++ che a rigore sono assenti dal linguaggio C, ma he possono essere ottenute anche con questo linguaggio al costo di qualche evitabile complicazione.

Si è preferito fare ricorso a C++ invece che al iù semplice C per varie ragioni.
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 al programmatore di controllare dei dettagli gli effetti delle operazioni effettuate nelle esecuzioni.
Questo lo consideriamo importante perché le caratteristiche delle operazioni implemntate sono in stretta corrispondenza con le caratteristiche delle operazioni riguardanti le strutture della matematica discreta presentati nei primi capitoli del tomo B e nel tomo D.

C++ consente anche commenti da collocare a destra della coppia di segni "\\" che sono più elastici dei commenti nei programmi in C; facilita il controllo di letture e scritture di caratteri grazie ai comandi `cin >>` e `cout <<`, utilizzabili sia in modo semplice che con versatilità. Esso 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 (non contemplata in c++), anche se possiede notevoli potenzialità per la implementazione di strutture matematiche e in particolare per elaborazioni su figure geometriche.

J11 0.03 In questo capitolo J11 di c++ vengono introdotto solo le 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 gradualmente 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 di calcolo della matematica.

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 in senso lato e delle conseguenti prospettive.
Queste, ampie e ambiziose, riguardano le possibilità di ottenere soluzioni condivisibili e di portata generale di problemi piuttosto pressanti 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 forniscono esempi significativi per la *esposizione*.

J11 0.04 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 J11 e J12 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 basilari per la matematica discreta, l'analisi numerica e la statistica.

Questi due capitoli non pretendono invece di bastare per preparare alla programmazione professionale 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 J13 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*.

Alberto Marini

J11 a. linguaggio C++, motivazioni e caratteristiche generali

J11 y.01 C++ è un linguaggio di programmazione, ossia un linguaggio artificiale che consente di scrivere testi leggibili normalmente, ciascuno dei quali ha lo scopo di richedere ad opportuni computers di eseguire elaborazioni finalizzate alla ricerca della soluzione di una certa gamma ben definita di problemi.

Più precisamente C++ è un linguaggio "general purpose", ossia un linguaggiodi programmazione con il quale si può cercare di richiedere ogni genere di elaborazione automatica; è inoltre imperativo, cioè consente soprattutto di esprimere comandi eseguibili automaticamente, ed à un linguaggio da compilare.
Quest'ultimo termine dice che ogni programma scritto in un tale linguaggio, per essere operativo, cioè per governare una elaborazione automatica effettuata da un adeguato computer, deve essere sottoposto preliminarmente a una traduzione in un programma eseguibile, prrogramma scritto in un linguaggio comprensibile alla macchina.
Va subito detto che con la compilazione si possono avere i programmi eseguibili più efficienti.

C++ è stato sviluppato da Bjarne Stroustrup nel 1983 come estensione del linguaggi C per ampliare le sue prestazioni, soprattutto per consentire la programmazione orientata agli oggetti come vedremo meglio in :d.
C++ è anche un linguaggio multiparadigma, in quanto riesce a supportare, oltre alla programmazione imperativa procedurale a diversi livelli di dettaglio, già consentita dal linguaggio C, anche gli stili della programmazione funzionale e della programmazione generica.

J11 a.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*.

C++ è 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 e sperimentino piccoli programmi con i quali possono prendere visione di esempi particolari di oggetti o processi matematici che vogliono conoscere meglio dopo averne apprese la definizione formale 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 ma non implementabili.

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.

J11 a.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.

J11 a.03 La scelta di c++ intende anche stimolare la costituzione di librerie di programmi che portino anche a considerare come problema culturale di rilievo la effettiva 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 sequenze finite, a partire da quelle su interi e stringhe, piuttosto che insiemi finiti.
È significativo e utile esaminare le differenze tra alcune formule matematiche e le loro possibili implementazioni.

J11 a.04 Terza motivazione: C++ si dimostra adatto a concretizzare le procedure per sostenere la redazione di questa stessa *esposizione* che sono state presentate in A05 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 alcune realizzazione nell'area della intelligenza artificiale (AI) stanno aprendo prospettive rilevanti sulle quali arà opportuno ritornare con più estese considerazioni.

J11 a.05 È opportuno anche segnalare che le attività di implementazione di algoritmi di interesse per la matematica conducono naturalmente a porsi problemi e a cercare approfondimenti sopra nozioni presenti da tempo in questa millenaria 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.

J11 a.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 sostiene che 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.

J11 b. primi esempi di programmi in C++

J11 b.01 Procediamo con la presentazione delle prime prestazioni del linguaggio C++ attraverso una progressione di programmi molto semplici.

Cerchiamo di mostrare esempi di 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 rendono disponibili per dare qualche tipo di soluzione ai problemi proposti.

I programmi iniziali richiedono solo una console costituita da una tastiera per l'immissione di richieste e un dispositivo di uscita per l'emissione delle risposte del computer; attualmente il dispositivo di uscita più comune è uno schermo monitor, ma si avrebbero comportamenti poco diversi se si disponesse solo di una stampante.
Nel passato un semplice utilizzo del computer si effettuava operando con una telescivente.

J11 b.02 Il primo tipo di programma che viene presentato, sostanzialmente in tutte le introduzioni, si limita a fornire una informazione elementare.

Il nostro primo programma intende solo presentare 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, senza dover entrare nei loro dettagli costitutivi e operativi.

Su questi meccanismi torneremo più avanti; per ora ci basta segnalare che la prima linea appartiene al tipo "direttive", e rende disponibili al programma vari meccanismi per le operazioni di ingresso e uscita (I/O); la seconda invece consente di disporre dei nomi che fanno parte delle collezione individuata da `std` appartenente al genere `namespace`; questa collezione la incontreremo spesso ed è assai estesa e variegata.

Viene poi l'intestazione del programma, `int main()`, che vedremo comparire all'inizio di ogni programma; altre intestazioni simili le troveremo all'inizio tutte le unità operative del linguaggio chiamate "functions".

Si incontra poi il cosiddetto "corpo del programma" delimitato tra i due segni coniugati `{` e `}`.
Nel corpo di ogni programma sorgente si individuano quelle che chiamiamo **frasi**, stringhe chiuse dal segno "`;`" che esprimono una ben definita richiesta o ordine; come sinonimo di frase si usano anche "enunciato" e "statement".

La prima frase del corpo del nostro programma comporta la emissione su quella che funge da console di uscita del sistema (schermo video, stampante o 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 direttiva che compare nella prima linea: "`<<`" precede la scrittura che viene emessa che nel linguaggio ha il ruolo chiamato "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.

Per maggiore completezza diciamo che `return` restituisce il controllo al sistema operativo del computer, ossia all'apparato software che si occupa del controllo generale delle attività del computer e che ora ci basta considerare come il meccanismo che presiede alla esecuzione dei programmi che vengono successivamente richiesti al computer da uno o più utenti.

Tra i possibili programmi qui ne 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 stringhe sue componenti che vengono terminate da una occorrenza di ";" .
Anche su leggibilità e frasi avremo modo di ritornare.

J11 b.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 con informazioni non banali; suggerisce anche che il linguaggo consente di avere programmi con tante frasi che comportano altrettante azioni; nel caso attuale sono fornite informazioni che verosimilmente alcuni utenti considerano richiami di loro utilità.

Dopo aver ricordato che un computer ha la possibilità di maneggiare grandissime quantità di dati, il precedente programmino può far immaginare che gli odierni sistemi elettronici ci possono aiutare fornendoci grandi quantità di informazioni condivisibili.

Se inoltre si ricordano le prestazioni dei prodotti delle tecnologie della informazione e della comunicazione, si può convincere che le informazioni a gestione digitale possono essere inviate e scambiate in tempi brevissimi, anche a moltissimi altri sistemi posti in grado di colloquiare con il nostro e che si possono trovare in ogni punto della Terra e delle sue vicinanze.

Può anche essere utile anticipare che i programmi del computer consentono di riordinare e rielaborare in tanti modi grandi quantità di dati, può ricavarne indicatori statistici e traducendo in rappresentazioni grafiche.

J11 b.04 Sin da questo inizio possiamo prospettare che gli strumenti elettronici e digitali possono contribuire enormemente alle nostre conoscenze, alle noste valutazioni, a conseguenti decisioni e a successive azioni.
In particolare qui prospettiamo la gestione efficiente di conoscenze che possono essere contenute in 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 linea alla successiva, il passaggio da una pagina a quella dopo e la presentazione di tabelle di dati.

È quindi naturale che i linguaggi di programmazione (nati inizialmente per eseguire pesanti calcoli numerici) si sono attrezzati per trattare anche segni che possono riguardare frasi discorsive, formule, azioni e manovre tipografiche.

In effetti nelle frasi di scrittura viste sopra compare il digramma "`\n`" che conclude le scritture ed ha funzione di richiedere di avviare l'emissione di una linea successiva.

Questo è un esempio di "escape sequence", scrittura che si serve di più caratteri semplici per rappresentare richieste non corrispondenti a un segno visualizzato.

J11 b.05 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 "`\\`".
Altre escape sequences sono "`\t`" per ottenere una salto per tabulazione orizzontale e "`np`" per ottenere il passaggio a nuova pagina.

Per quanto riguarda la gestione automatica dei testi in generale occorre segnalare che si tratta di una problematica articolata 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 di veicolarli attraverso vari canali telematici, per non parlare di immagini, animazioni e archivi di conoscenze di ogni genere.

J11 b.06 Finora si sono trattati solo dati simbolici, singoli caratteri o stringhe, costanti, ossia informazioni che possono solo essere emesse.
Ora vogliamo trattare informazioni sia simboliche che numeriche che possono venire modificate da interventi di operazioni eseguite dai circuiti che si trovano nella "unità centrale" del computer, ossia nel suo centro operativo.

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 e vogliamo ripetere che tutte sono caratterizzate dall'essere concluse da ";".

La prima fa emettere una stringa leggibile seguita dalla emissione della richiesta di passaggio a capo espressa da `endl` (richiesta che 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 le due costatanti numeriche intere annunciate, richiede di sommarle, di emettere il risultato seguito da una parola di spiegazione e di effettuare 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 usuale ben nota della matematica e quindi una espressione con un significato operativo facilmente intuibile; in effetti ottiene l'emissione del numero risultante (1290) e il passaggio a nuova linea.
La quarta frase serve solo per farci sapere che la sottrazione viene rappresentata dal segno "-", fatto abbastanza prevedibile. e fa emettere la differenza risultante.

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 1290
18
```

J11 b.07 L'esempio precedente può essere facilmente generalizzato: dovrebbe essere chiaro come si possono ottenere i risultati di calcoli determinati da più operazioni aritmetiche sopra operandi interi forniti da scritture di costanti numeriche, e come quasti risultati possono essere alternati a scritture esplicative utili a chiarire i loro significati.

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; si chiede che il programma trovi 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; il lingotto è costituito da una lega avente la densità di 7 kg al decimetro cubo.

J11 b.08 C++, come tutti i linguaggi di programmazione per attività con esigenze quantitative (nelle scienze, nelle tecnologie, nelle costruzioni, nei commerci, nelle amministrazioni, ...) permettono di

operare su numeri razionali esprimibili finitamente con notazioni decimali i quali che si avvicinano ai numeri reali, informazioni chiamate numeri in floating point, in breve numeri-fp.

Alcuni esempi di scritture di numeri reali sono: 12.5, -1.6666, +9.81, 400000., -273.23, 259000.125, .75. Queste scritture prevedono un segno - necessario, un segno + opzionale, una parte intera che può mancare. un punto con il ruolo di separatore e una parte decimale che può mancare.
Non sono invece scritture 400000 che esprime un numero intero, e 3.1415..., che viene usato per indicare discorsivamente un unmero reale non razionale.

Le precedenti scritture permettono di individuare quantità poco lontane dall'unità; si possono però trattare anche numeri con valori assoluti molto maggiori di 10^{16} e molto inferiori a 10^{-16} con scritture che accanto alle precedenti pongono una potenza di 10 da considerare fattore moltiplicativi.

Si possono quindi manipolare numeri positivi e nagativi il cui valore assoluto varia circa da da 10^{70} a 10^{-70}.

Esempi: .....

J11 b.09 Le calcolatrici elettroniche programmabili consentono di ottenere attraverso una serie di calcoli i valori che fornisce una formula anche elaborata in corrispondenza a diverse scelte dei valori che possono assumere i suoi parametri.

È quindi ragionevole aspettarsi che ogni buon linguaggio di programmazione consenta di trattare informazioni riguardanti non solo dati indicati nei programmi stessi come costanti, ma anche dati ai quali in successive esecuzioni vengono assegnati direttamente in lettura o indirettamente valori diversi che esprimono le diverse circostanze che hanno motivato le diverse esecuzioni.

Dunque un linguaggio come C++ deve consentire il trattamento di dati variabili.

Queste sono entità rappresentati da identificatori ai quali il compilatore associa celle di memoria, le quali ospitano valori che possono cambiare nel corso di ciascuna delle possibili esecuzioni che vengono definite dai diversi possibili sistemi di dati iniziali.
—JQ Esempi ii identificatori per il linguaggio C++: `i, k2, pigrc, alt_cono, ricaviAnno2022, Richard_Feynman, sogliaMassima`.

Come suggerisce il gruppo dei precedenti esempi, si possono adottare identificatori concisi, comprensibili solo al programmatore (che potrebbe oure dimenticarne il significato), oppure identificatori di significato più evidente e menminico, facile da ricordare.
La forma degli identificatori, chiamati anche "nomi delle variabili", in C++ e in molti altri linguaggi, sono meno liberi di quelli dai linguaggi naturali, in quanto devono essere distinguibili agevolmente all'interno di un programma sorgente.
In particolare non possono comprendere spazi bianchi e segni come "-", "'" e apostrofo che nel linguaggio hanno loro compiti specifici.

J11 b.10 I valori delle possibili variabili possono variare in insiemi diversi ed essere trattati in modi diversi dalle operazioni eseguite da diversi circuiti dell'unità centrale che le frasi del programma possono richiedere.
Si distinguono quindi diversi tipi di variabili: sono disponibili variabili che possono assumere valori interi, variabili per numerici-fp, variabili che possono assumere come valori caratteri o stringhe e variabili che possono assumere solo i "valori booleani" vero e falso.

Ogni variabile viene introdotta nel programma con una frase del genere dichiarativo che definisce il tipo al quale appartiene; ciascuna di queste frasi è caratterizzata da una scrittura particolare chiamata **parola chiave**.

La parola chiave `int` riguarda i numeri interi come 123, -45 o 384902203, mentre `double` riguarda numeri reali fp, come 19.25 or -9.98.

Possiamo quindi segnalare come si possono introdurre alcune espressioni che riguardano quantità numeriche, esprimibili con numeri interi e reali ben note nella geometria, nella fisica e nalla statistica.

J11 b.11

La parola chiave `char` consente di introdurre variabili che possono riguardare singoli caratteri visualizzabili come `'a'`, `'W'` o `'+'` o non visualizzabili forniti da scritture del genere escape sequence.

`string` permette di dichiarare variabili i cui valori sono espressi da sequenze di caratteri delimitate da doppi apici come `"string"`, `"Decameron"` e `"testo"`.

`bool` consente di introdurre variabili ciascuna delle quali può assumere due valori vero o true e falso o false; tali variabili sono utilizzate in formule che esprimono condizioni che dipendono dal verificarsi o meno di eventi diversi.
Per esempio si abbiano a disposizione degli interpreti che possono conoscere o meno inglese, francese, tedesco, italiano, spagnolo, cinese, giapponese e coreano.
Tra questi professionisti occorre selezionare interpreti che mettano in collegamento una azienda canadese con una finanziaria svizzera; questi quindi devono conoscere il francese, oppure l'italiano, il francese e l'inglese, oppure il tedesco, il francese e l'inglese.

J11 b.12 secondo programma e sua spiegazione semplice con considerazioni sopra la necessità di commenti adeguati.

J11 c. costanti e variabili; dichiarazioni e assegnazioni

J11 c.01 Per quanto detto, un programma per un computer dotato di un semplice corredo di unità periferiche (tastiera, stampante, monitor video, porta USB per memorie a stato solido), può vedersi come un complesso di richieste espresse secondo regole formali piuttosto precise che, ogni volta che viene presentato al computer con un adeguato complesso di dati di ingresso (numerici, simbolici, logici o di altri generi non ancora accennati) governa l'esecuzione di una sequenza di operazioni la quale, se non si verificano situazioni da considerare patologiche, fornisce nuovi dati che rivestono interesse quando costituiscono i risultati di una elaborazione con il fine di trovare una risposta a un problema.

Ogni esecuzione di un programma può elaborare vari generi di dati: oltre ai dati forniti in ingresso, acquisiti in seguito a frasi di lettura, possono essere elaborati dati espressi nelle frasi del programma e dati che le manovre richieste dal programma hanno prodotto in precedenti fasi dell'esecuzione; in genere solo una piccola parte di tutti i dati trattati 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**.

J11 c.02 Un computer attuale, per effettuare le elaborazioni che ci si aspetta possano essergli richieste, deve essere dotato di strumenti operativi che hanno la forma di programmi predisposti e dati registati in suoi dispositivi di memoria; tra queste memorie si possono distinguere le memorie centrali e le memorie periferiche; prevalentemente dischi fisici o simulati su dispositivi a stato solido; tra queste ultime si distinguono quelle che fanno parte permanente del computer da quelle asportabili (dischi esterni e pen drives).
Le memorie centrali sono accessibili più rapidamente, le periferiche possono essere più estese.
Contrariamente al passato, attualmente sono disponibili estese memorie a stato solido accessibili molto rapidamente e il software di base che sostiene i programmi in linguaggi come C++ consente al programmatore di gestire grandi quantità di memoria senza troppo preoccuparsi della loro collocazione fisica, in quanto le collocate perifericamente può vederle senza distinguerle dalle centrali.

Parte dei programmi di base del computer servono a renderlo funzionante ed efficiente il computer e fanno parte del suo "sistema operativo".
Altri programmi predisposti riguardano esigenze che si presume possano essere richieste da molti dei programmi che si prevede saranni proposti dai possibili utenti.

Questi ultimi, naturalmente son chiamati **programmi degli utenti**, user programs, mentre i programmi predisposti per effettuare manovre di interesse generale, sono sottoprogrammi organizzati in collezioni dette "librerie di programmi".

Il complesso dei programmi predisposti attualmente è molto articolato, ma l'utente interessato ai risultati non deve approfondire i suoi dettagli, deve conoscere soltanto quanto serve per richiamarli per ottenere le prestazioni utili per i propri obiettivi.

J11 c.03 Una parte primaria degli strumenti in dotazione del computer costituisce Il sistema operativo del computer (Windows, Unix, Mac OS, ...) e comprende molti programmi che interagiscono con l'utente al fine di fornirgli tutti i supporti che gli sono necessari per consentirgli di rendere esecutive tutte le procedure che riguardano direttamente le tasformazioni dei suoi dati nei risultati che persegue.

Per consentire che i suoi utenti ottengano con efficacia e con facilità le esecuzioni governate dai programmi applicativi, il computer deve essere dotato anche di un sistema di programmi anch'essi decisamente complessi che chiamiamo **sistema di sviluppo**.

Un sistema di sviluppo ha due scopi principali: (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.

J11 c.04 Anche il sistema di sviluppo per 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 prendere facilmente le decisioni più opportune per il raggiungimento dei suoi obiettivi.

La visione del sistema di sviluppo deve essere tanto più estesa quanto più sono impegnative ed estese le attività di programmazione, di conduzione delle elaborazioni, di organizzazione dei risultati e di previsione di cambiamenti evolutivi.

L'acquisizione di una visione di questa ampiezza da parte di un utente abituale del computer in genere si configura come formazione di un proprio metodo di lavoro e di un proprio stile di scrittura dei programmi.

Questa formazione risulta particolarmente impegnativa per le problematiche che richiedono intere squadre di programmatori che devono riuscire a padroneggiare le varie tecniche che si devono applicare per mettere a punto le molteplici procedure risolutive che devono essere adottate per ottenere risultati soddisfacenti.

J11 c.05 Ogni sistema di sviluppo di un linguaggio di ampia portata è un prodotto industriale decisamente evoluto che contiene componenti che in buona parte dipendono 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 tutti gli errori lessicali e sintattici che sa trovare nel testo del programma e gli errori semantici potrebbe trovare o sospettare nella organizzazione del programma;
- segnalare le eventuali anomalie che riesce a riscontrare nel corso dell'esecuzione (anche in seguito a richieste espresse dal programmatore;
- mettere a disposizione ampie librerie di sottoprogrammi di utilità generale;
- supportare la gestione di programmi di sistema con i quali il programma scritto dall'utente può interagire;
- aiutare il programmatore nella individuazione dei propri errori e delle ambiguità nelle sue richieste.

Dopo il precedente semplice accenno, ritorneremo su alcuni aspetti del sistema di sviluppo per chiarire questioni che si andranno ponendo in relazione a specifiche prestazioni del linguaggio.

J11 c.06 Torniamo a considerare le celle di memoria nelle quali vengono registrate le informazioni che un programma elabora, ovvero i valori che ciascuno dei dati variabili che riguardano la risoluzione del problema trattato viene ad assumere nelle successive fasi di una esecuzione;

Il valore di un dato registrato nella sua cella in ciascuna delle fasi esecutive viene detto **valore corrente** del dato.

Senza entrare nelle tecniche adottate dal traduttore per associare l'identificatore di una variabile o di una costante alla sua cella, tecniche che possono essere diverse per i diversi sistemi operativi, possiamo pensare che il traduttore organizzi una tabella che associa a ogni identificatore di dato che compare in un programma la corrispondente collocazione in una cella o in una zona della memoria consistente in una sequenza di celle.

La memoria del computer dal punto di vista di un programma si può considerare come un nastro nel quale si distinguono successive sequenze di celle dedicate ad aggregati di dati tendenzialmente omogenei.

Ricordiamo anche che si possono avere celle di varie taglie: bytes per caratteri e per i valori booleani, celle di diverse estensioni per gli interi e per i numeri-fp.

J11 c.07 Ciascuna delle celle 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 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 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, e quindi per il 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 che il programmatore poteva scegliere in modo da poterli ricordare con una certa facilità.

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

J11 c.08 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 dell'anno 1950 ai miliardi di esemplari attuali, si è trasformato dall'essere una pratica guidata dalle esigenze di applicazioni specifiche e dalle caratteristiche di singoli prodotti hardware al costituire una fenomenologia complessa da far evolvere in relazione all'andamento dell'economia e delle imprese, alle esigenze di ampie categorie di utilizzatori più o meno diretti e allo sviluppo della società; questo ha condotto a una nuova cultura

che dà ampio credito alle metodologie del trattamento delle informazioni e alle soluzioni condivise di molti problemi.

Per l'evoluzione del tattamento delle informazioni non si possono trascurare le influenze dei detentori di poteri riguardanti imprese finanziare, industriali, della logistica e dei commerci, dei media e dello spettacolo, dei servizi sanitari e dell'istruzione, i poteri militari e politici, nonché, indirettamente ma massicciamente, la vasta schiera dei consumatori di intrattenimento e di videogiochi.

Naturalmente non ci addentreremo nei molteplici aspetti della società odierna, ma ci poniamo dal punto di vista della adozione di un linguaggio di programmazione procedurale medio-alto e di portata vasta come il C++, ritenendo che la segnalazione dei collegamenti più influenti tra i suaccennati sviluppi e le metodologie della programmazione possa servire a chiarire molte scelte che hanno portato a modi di procedere dei costruttori e degli utenti dei computers.

J11 c.09 In un linguaggio come C++ i dati costanti sono individuati mediante scritture che seguono precise regole 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 per essere adattato in tempi successivi al suo primo utilizzo, soprattutto quando si prevede che su esso possa destare l'interesse di nuove attività.

Il sistema di sviluppo del linguaggio si incarica di assegnare a tutti i dati elaborati da un programma le opportune celle. In ogni cella destinata a un dato costante inserisce il valore espresso dalla scrittura di costante che introduce tale dato.
Anche le celle per i dati variabili hanno un contenuto iniziale, ma è possibile che questi valori siano decisi diversamente dai diversi sistemi operativi istallati sui diversi computers.
Per evitare rischi e per avere programmi più trasferibili il programmatore per ogni ogni dato variabile deve preoccuparsi che il dato sia inizializzato correttamente da una richiesta di lettura o da una assegnazione; questerichieste esplicite possono essere dettate anche dalla sola prudenza.

Solo per programmi per esigenze molto specifiche è necessario addentrarsi nei dettagli dei collegamenti tra gli identificatori delle variabili e le celle loro assegnate; questi dettagli dipendono dal comportamento interno del sistema di sviluppo.
Per la maggior 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 i suddetti collegamenti nelle fasi esecutive.

Le tabelle di collegamento identificatori-indirizzi fanno sì che una elaborazione che gestisce i dati attraverso i loro identificatori risulti equivalente a una elaborazione che gestisce i dati servendosi degli indirizzi per la memoria centrale (con il linguaggio delle macchina) o degli indirizzi per memorie su nastri (come con la macchina di Turing).

J11 c.10 Veniamo alle regole per il controllo dei dati con il linguaggio C++.

Diciamo **alfabeto degli identificatori** l'insieme costituito dai caratteri alfabetici minuscoli e maiuscoli, 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, tolleranza;
```

Veniamo alla scelta degli identificatori per un programma che prevedibilmente dovrà essere riadattato a varie successive richieste applicative.

In linea di massima il programmatore deve scegliere tra due esigenze che possono trovarsi 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.

J11 c.11 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 di estensioni diverse che ovviamente devono essere più estese per valori assoluti maggiori.

Un intero grande esige una cella estesa, ma si può chiedere di inserire un intero piccolo in una cella estesa.
Se ad esempio si vuole introdurre una costante intera da assegnare a una cella-64b, cioè se si vuole una costante del tipo associato alla parola chiave `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 esprimendo tale numero intero con la scrittura `12L`.

J11 c.12 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 delimitare con 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	\\	\”

J11 c.13 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, progrScolari, 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** del linguaggio C++, 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 e di predisporre che i rispettivi contenuti nel corso della esecuzione siano trattati come entità di un tipo ben definito, nei casi attuali 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 spesso sono 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**.

J11 c.14 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; conseguentemente il compilatore fissa l'indirizzo di tale cella che il programmatore in genere può evitare di conoscere, in quanto può controllare il dato servendosi 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 compatte; le precedenti frasi si possono sostituire con la sola seguente:

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

J11 c.15 In generale si possono unificare le frasi dichiarative e le frasi di inizializzazione che riguardano le stesse variabili.
Invece delle cinque frasi riguardanti variabili intere di :c13, supposto che le frasi di assegnazione siano frasi di inizializzazione, si possono sostituire con le seguenti:

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

```
short matrIscritti,numIscritti=102,pergScolari,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`.

J11 d. arrays e stringhe

J11 d.01 Per affrontare moltissimi problemi si rende necessario costruire ed elaborare collezioni di dati omogenei, innanzi tutto sequenze di questi dati.

Sequenze di interi consentono di fornire i numeri degli abitanti di una sequenza di città o di nazioni, più in generale le numerosità (ossia i cardinali) degli insiemi presentati con un loro elenco, le altitudini in metri di località incontrate lungo un certo percorso o presentate in ordine alfabetico, le quantità di un tipo di oggetto prodotto in anni successivi, i punteggi delle squadre partecipanti a un campionato.

Una sequenza di numeri naturali può servire a registrare misurazioni di grandezze ottenute in osservazioni successive come lunghezze, pesi, durate, velocità, quantità di denaro, temperature di un paziente, concentrazioni, percentuali, indici di gradimento, ...

Oltre alle sequenze numeriche, risultano palesemente utili le sequenze di caratteri; esempi di queste sequenze sono i caratteri che si incontrano in un nome assegnato a una qualsiasi entità, nella denominazione di una località o di una persona, in un verso poetico, nel simbolo di un composto chimico, in un acronimo, nel codice ISBN che consente di individuare un libro, ...

Il modo canonico per registrare per un computer 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 assimilare alla registrazione di dati omogenei sopra un segmento di un nastro di carta, di un nastro magnetico e di una traccia di un disco magneto-ottico o di un nastro di una astratta macchina di Turing.

Come si è detto, ogni sequenza di celle 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 e dal numero di celle sulle quali si deve avanzare partendo dalla iniziale per giungere a quella che interessa.

J11 d.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 collezioni di dati che consentono al programmatore di accedere facilmente alle loro componenti, ossia senza preoccuparsi della loro precisa collocazione nella memoria fisica, in quanto di questa si fa carico il sistema traduttore dei programmi.

Con C++ si possono trattare arrays di una, due o più dimensioni, strutture che complesivamente qui chiamiamo "schieramenti di dati".
Per 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 denomL,incassiN;
int incassi[50];
```

La prima frase stabilisce che servendosi dell'identificatore tt denom si possono trattare denominazioni di al più 25 caratteri il cui numero in ciascuna delle circostanze è determinato dal valore attuale della variabile intera che introduciamo con la frase successiva identificandola con `denomL`.

La terza frase chiede di controllare attraverso l'identificatore `incassi` sequenze di al più 50 interi il cui numero attuale è ragionevole supporre che sarà contenuto nella variabile `incassiN` che compare dopo `denonL`.

Nei programmi che presentiamo si assume che i valori attuali dei caratteri siano elementi dell'alfabeto ASCII (per la maggior parte visualizzabili, come viene riciesto nella gran parte delle applicazioni pratiche).
Inoltre, in mancanza di richieste esplicite, si assume e che ciascuno degli interi sia registrato in 32 bits consecutivi e quindi possa assumere tutti i 2^{32} valori appartenenti all'intervallo $[-2^{31} : 2^{31} - 1]$.

J11 d.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';
```

Ciascuna di queste frasi esprime richieste operative con conseguenze simili: ciascuna di esse ha l'effetto di determinare il valore 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-8b determinata dalla scrittura alla sua sinistra.

Conviene segnalare anche qui che il segno "`=`" non esprime una relazione di uguaglianza e non caratterizza una equazione, come fa quando compare in gran parte delle formula matematiche; esso invece richiede una azione consistente nella assegnazione di un nuovo valore ad una variabile, cioè nella modifica del contenuto di una cella.
Possiamo dire che "`=`" esprime una operazione di assegnazione e va assimilato al connettivo "`:=`" usato spesso quando si definisce una entità matematica e adottato talora per introdurre in un discorso un nuovo termine che si intende usare con un suo preciso significato.

J11 d.04 La prima delle precedenti frasi di assegnazione comporta la determinazione della rappresentazione dell'intero 9, costituita dalla sequenza di 32 bits `0000 0000 0000 0000 0000 0000 0000 1001`, 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 secondo il codice ASCII del carattere "`u`", costituita dall'ottetto `10101110`, e l'inserimento di tale ottetto nel byte costituente la 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; questa evidentemente è accessibile attraverso l'espressione `denom[0]` .

Osserviamo 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 in un linguaggio come C++ ([0], [1], [2], ..., [$n-1$], ...).

Conviene anticipare che l'indicazione che segue l'identificatore di un array per individuare un suo componente va interpretata come espressione di una azione piuttosto che come precisazione statica.
Una scrittura come `denom[4]` comporta la valutazione di un valore numerico (in questo esempio abbiamo subito 4, ma tra le parentesi quadre si potrebbe avere anche una espressione aritmetica piuttosto elaborata), seguita dall'incremento dato da tale valore diminuito di 1 all'indirizzo iniziale dell'array.
Questo indirizzo è misurato in bytes per `denom`, mentre va misurato in quaterne di bytes per un array di tipo `int` e con criteri simili per le celle di altri tipi di dati aventi altre estensioni.

J11 d.05 Se vogliamo disporre dei primi 10 numeri della successione di Fibonacci (wi) nelle prime 10 componenti dell'array che ha come identificatore Fib , possiamo 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;
```

Si osserva che il segno ";" che chiude tutte le frasi, svolge un ruolo di separatore di frasi ben riconoscibile dal programmatore e dal compilatore; ciascuna di queste frasi comporta l'assegnazione del valore fornito dalla scrittura decimale al secondo membro alla cella associata alla data componente di array; le successive frasi 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 più facilmente controllabile:

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

J11 d.06 Abbiamo visto dunque un esempio di dichiarazione e inizializzazione di un intero array di interi.
Una frase di questo tipo riguardante un qualsiasi array monodimensionale 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 nel suo complesso delimitata da parentesi graffe.

(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 ai valori interi dei 20 elementi chimici più leggeri.

J11 d.07 Passiamo a considerare Le sequenze di caratteri, cioè le stringhe ricordando che servono per affrontare molti problemi.

Innanzi tutto le stringhe di caratteri visualizzabili sono necessarie per tutte le elaborazioni di informazioni esprimibili nei linguaggi naturali: nomi e qualificazioni di persone, di prodotti e di concetti di ogni genere, citazioni letterarie, slogan, norme, ricette, tabelle di dati, trascrizioni di registrazioni, testi normativi, legislativi, poetici, narrativi, ...

Ogni testo scritto o comunque registrato può essere ridotto a una stringa (eventualmente ricorrendo alla collezione Unicode [:d05, :d06].
Molte importanti informazioni vengono espresse mediante linguaggi convenzionali e artificiali: espressioni matematiche, enunciati della logica, formule chimiche, sequenze biologiche, norme d'uso di apparecchiature, leggi, ...

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 di linguaggi per questi scopi: TEX (we), HTML (we), XML (we) ed SVG (we).

Mediante stringhe si possono esprimere anche testi altamente complessi come spartiti musicali, partiture orchestrali e registrazioni sonore, tutti trasformabili in sequenze di bits per le quali si possono prospettare elaborazioni rette da programmi di rilevante utilità.

J11 d.08 Riconosciuta appieno l'importanza applicativa delle stringhe, i linguaggi C e C++ hanno reso disponibili vari strumenti per il loro trattamento.

Le prime componenti dei programmi che abbiamo incontrato [] sono gli **string literals**, le scritture che consentono di definire le stringhe costanti 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 righe 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 righe del file .txt che fa \
da supporto per il testo davanti ai loro occhi."
```

In uno string literal possono comparire utilmente 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 consente la definizione e 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 in genere viene collocato in una sequenza di bytes occupati dai successivi caratteri presentati tra i doppi apici seguiti da un ottetto contenente il carattere null, il primo dell'elenco dei caratteri ASCII costituito da otto bit uguali a 0.
Le stringhe con un null finale si possono elaborare con modalità omogenee ben sperimentate sono chiamate **null teminated strings**.

J11 d.09 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, ma si sarebbe dovuto 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 $\mathbf{[}2^{31} : 2^{31}-1\mathbf{]}$, in quanto solo i primi 8 numeri sono trattabili con queste celle; infatti l'ottavo è $M_{31} = 2^{31}-1$, mentre il nono, $M_{61} =$ `2305843009213693951`, e il decimo, $M_{89} =$ `618970019642690137449562111`, richiedono più di 32 bits.

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 con i numeri-fp, i quali però non sono in grado di garantire completa precisione e quindi di trattare con facilità vari tipi di elaborazioni numeriche; ad esempio non sono sensatamente utilizzabili per trattare problemi di natura crittografica.

Ribadiamo che i limiti finiti degli strumenti effettivi, ovviamente, devono essere tenuti ben presenti in tutte le attività computazionali non elementari.

Tali limitazioni nello sviluppo di considerazioni generali, possono invece essere considerate un intralcio alla formulazione rapidamente comprensibile di molti enunciati e di molte argomentazioni.
In effetti gli atteggiamenti di chi porta avanti studi matematici specialistici e gli atteggiamenti di chi si occupa di calcoli specifici possono essere assai piuttosto diversi e in certe questioni richiedono di adottare accorgimenti discordanti.

J11 f. operazioni di entrata e uscita

J11 f.01

Comdideriamo le seguenti frasi:

```
int x;
const pigr 3.14 ;
cout << "Digita il valore del raggio della sfera:  "; // richiesta iniziale
cin >> x; // ottiene dalla tastiera della console il valore richiesto
// calcola volume e superficie della sfera e li emette
cout << "sfera:  volume = " << 4.55*x^3 << " superficie = " << 6.28*x^2 << endl;
```