1

Capitolo B80
elaborazioni automatiche e programmazione

Contenuti delle sezioni

24 pagine

B800.01 Questo capitolo introduce a grandi linee varie nozioni riguardanti il mondo della elaborazione automatica delle informazioni e delle macchine che consentono di realizzarlo, i computers.
Si inizia con una panoramica slle idee che si avevano sul calcolo nel passato e sugli li strumenti che consentivano di facilitarlo.
Successivamente si percorre l'evoluzione degli strumenti elettronici per il calcolo automatico e gli sviluppi delle numerose attività che l'hanno accompagnata.
Viene in seguito discussa, sempre per grandi linee, l'evoluzione delle attività di programmazione.
Segue una carrellata sulla varietà dei sistemi hw e sw che sono oggi disponibili e nei quali la programmazione svolge sempre un ruolo primario.
Nell'ultima parte si esaminano i vari tipi di dati che si possono elaborare automaticamente e le modalità per le loro rappresentazioni ai diversi livelli dell'hardware, della programmazione e delle applicazioni.

B80 a. strumenti di calcolo del passato

B80a.01 Nel passato: meccanismo di Anticitera (calcolatore astronomico), macchina di Schickard (orologio calcolatore), calcolatori aritmetici di Pascal e di Leibnitz, Thomas da Colmar e altre calcolatrici meccaniche basate sulla meccanica di precisione, calcolatrici elettromeccaniche fino alla Coptometer), Borroughs, macchine a schede (le prime per il controllo dei telai Bouchin-Falcon e fino al telaio di Jacquard) di Hollerith e di Thomas Watson (IBM)
Grande progetto di Babbage con Ada Lovelace.
Filone dei calcolatori analogici: compasso di proporzione, regolo calcolatore, macchine integratrici e macchine derivatrici e il computer analogico analizzatore differenziale meccanici di Vannevar Bush.
Si serve digrandezze fisiche meccaniche, idrauliche o elettriche per esprimere quantità idealmente esprimibili come numeri reali.

B80a.02 I primi computers
ABC
Stibitz
Zuse
Bletchley Park ed Enigma
ENIAC ...
Architettura di Von Neumann ispirata da Turing

B80 b. panoramica della crescita dei computers

B80b.01 Dalla fine dalla WWII ad oggi

B80b.02 Le prime macchine costruite negli ultimi anni 1940 erano prodotti unici costruiti da piccoli gruppi di ricercatori che comprendevano le persone interessate a calcoli numerici particolari.
Questi gruppi di sviluppatori si dovevano costruire quasi tutti i componenti della macchina e gran parte degli attrezzi necessari per realizzarla.
Quindi queste macchine venivano controllate con metodi peculiari e fornivano risultati di interesse corcoscritto, quasi esclusivamente scientifico e militare.

Negli anni 1950 si sono resi disponibili nuovi tipi di dispositivi hardware (memorie a nuclei, nuovi lettori di nastri e schede di carta, tamburi enastri magnetici e alla fine componenti logiche a stato solido che hanno rimpiazzato le valvole termoelettroniche), si sono ampliati i metodi di calcolo (analisi numerica, ricerca operativa, programmazione dinamica, ...), si sono chiarite molte conoscenze fondamentali (teoria dei linguaggi e della computabilità, progetto della intelligenza artificiale) e si è consolidata la previsione di una vasta gamma di applicazioni nell'industria, nei commerci e nelle amministrazioni) e quindi della necessità di sviluppare solide basi industriali, finanziarie e universitarie per il settore dei computers.

In questi anni nascono i primi linguaggi di programmazione che intendono essere indipendenti dalle macchine, Fortran (Backus e Naur) e Cobol (Grace Hopper).

Inoltre sono nati i primi modelli di computers, chiamati mainframes, da costriure in serie.

B80b.03 Negli anni 1960 si è giunta a maturazione la teoria dei linguaggi di programmazione e della compilazione e si sono definiti sistemi operativi in grado di facilitare la ampia utilizzazione del calcolo automatico.
Da una situazione di quasi monopolio della IBM si è passati alla presenza di varie industrie del computers e a una differenziazione delle macchine. (serie di macchine di grandi dimensioni, computers per impianti industriali, embedded computers, ...).
Nell'ambito della programmazione si sono imposti nuovi metodi (programmazione strutturata, attenzione verso il ciclo di vita del software, Sistemi per la gestione delle basi di dati, ...), si è consolidata l'ingegneria del software, si sono poste le basi per la standardizzazione dei linguaggi di programmazione e di altri linguaggi artificiali.

B80b.04 Negli anni 1970 arrivano i minicomputers per grandi numeri di piccoli ambienti produttivi, amministrativi e universitari.
Cresce lo studio dei sistemi operativi, in particolare nasce UNIX e con esso il linguaggio C.
Si impongono i microprocessori e i video terminali e si giunge ai personal computers che iniziano a diffondersi capillarmente.
Giungono le prime workstations con elevate prestazioni di calcolo , anche simbolico.
Un forte elemento propulsore sono i videogiochi (Atari Commodore.

B80b.05 Negli anni 1980 cresce il filone dei supercomputers.
Nell'ambito dei linguaggi di programmazione nasce l'idea della programmazione orientata agli oggetti (Algol 68, Simula).
In questa direzione Stroustrup definisce C++ come evoluzione del linguaggio C che amplia tutta la sua portata, ma presenta alcune differenze sintattiche rese opportune dal fatto che si serve di librerie standard molto più ampie e dall'essere in progressiva evoluzione.

Cresce Internet ecrescono le attività che vedono la cooperazione di computers distanti collegati da strumenti telematici che diventano sempre più necessari.
Si risolleva l'intelligenza artificiale dopo uno dei suoi "'inverni", spinta dal piano giapponrse deutto dell aquinta generazione. Contemporaneamente rinasce il progetto dei circuiti neuronali artificiali.

B80b.06 Gli anni 1990 principalmente vedono la nascita del WWW, con la definizione di HTML da parte di Tim Brners Lee presso il CERN.; nel 1997 vengono decisi grandi investimenti nelle iniziative basate su Internet da parte di varie industrie informatiche; vengono poste la basi per i linguaggi per Internet soprattutto ad opera del consorzio W3C; nascono i primo web brousers, nasce il linguaggio Java.
Vengono definiti numerosi sistemi per la programmazione della Rete globale e inizia la crescita dei siti sui Web; si diffondono i laptops. Le attività ludiche digitali vedono crescere vistosamente i praticanti.
L'intelligenza artificiale diventa popolare grazie alla vittoria sul campione Garry Kasparov da parte del sistema Deep Blue di IBM nel 1996.

B80b.07 Nel primo decennio del 2000 nasce l'economia su Web, cresce la globalizzazione. Nel 2004 nasce Wikipedia e in pochi anni diventa l'opera enciclopedica internazionale di gran lunga maggiore.
So sviluppano grandi archivi su Web e cresce i movimenti dell'open content e dell'open source; sono lanciati progetti di conoscenze condivise come progetto Gutemberg, arXiv e GIMPS = Great Internet Mersenne Prime Search.
Sono disponibili sempre più numerosi testi liberi, in particolare tutorials.

Nel 2007 inizia la diffusione degli smart phones e quindi accelerando vistosamente l'uso della rete e la globalizzazione.
Si sviluppano i social media e la loro influenza su numerosi aspetti della vita, sia al livello dei comportamenti individuali, sia al livello delle attività industriali e amministratibve, sia al livello delle economie nazionali.

B80b.08 Negli anni 2010 emergono con prepotenza le prospettive dell'intelligenza artificiale. Con la vittoria nel gioco Go del sistema AlphaGo della Google DeepMind sul campione Lee Se-doli nel 2016 si consolida la fiducia nell'intelligenza artificiale.

B80b.09 All'inizio del decennio 2020 importanza dell'informatica nel contrasto alla pandemia Covid-19, nella disinformazione, nel controllo delle popolazioni, nelle attività belliche.
Nel 2022 si impone con ChatGPT di OpenAI l'AI generativa dei Large Language Models. Si rendono disponibili buoni traduttori automatici.
Si impongono le problematiche della Artificial General Intelligence, del superumanesimo, dei pericoli derivanti dall'uso non regolamentato dalla AI.
Crescono gli strumenti AI per biologia, farmacologia, scienze varie.

B80 c. programmazione dei computers, generalità

B80c.01 La programmazione degli elaboratori elettronici è oggi una delle attività più importanti da tanti punti di vista, fatto evidenziato dalla onnipresenza di questi strumenti.

L'elaborazione automatica ha suscitato interessi economici, culturali e politici che toccano, più o meno direttamente, tutti gli aspetti della vita dei contemporanei.

L'efficienzae l'efficacia nelle attività di programmazione condizionano l'efficienza e la efficacia di interi settori amministrativi, commerciali e industriali e risultano determinanti per la capacità di interi sistemi paese di riuscire ad evolversi nell'odierno panorama di cambiamenti continui e generali.

B80c.02 Esaminiamo brevemente gli aspetti del mondo della programmazione.

I dispositivi per il calcolo elettronico di maggior rilievo, quelli che chiamiamo computers, hanno la possibilità di adattarsi ai cambiamenti degli ambienti nei quali sono inseriti prendendo delle decisioni a partire dagli stimoli che giungono dal loro intorno o da ordini che provengono da un loro controllore presente o remoto.

Queste decisioni sono effettuate mediante elaborazioni che dall'esterno sono viste come trasformazioni di informazioni di ingresso in informazioni di uscita, trasformazioni effettuate da operazioni realizzate da loro circuiti organizzate da programmi per i computers predisposti al loro interno.

Una elaborazione è costituita da una sequenza di passi che vedono successivi cambiamenti della configurazione del computer e che consistono in piccoli cambiamenti delle informazioni contenute nei dispositivi di memoria del computer stesso e che nel complesso portano alla derivazione dalle informazioni in ingresso, i dati della elaborazione attuale, delle informazioni in uscita, i risultati della elaborazione.

B80c.03 Ogni elaborazione viene controllata da un programma che consideriamo una implementazione, ossia un adattamento al computer, di un algoritmo che può collocarsi a livelli di complessità molto diversi.

Anche un programma è costituito da un complesso di informazioni che può essere registrato nel computer e trasmesso attraverso canali di comunicazione tra dispositivi digitali anche molto distanti tra di loro.

Ogni programma viene trattato da persone, che qui chiamiamo programmatori, da dispositivi di calcolo, i computers, e da apparecchiature per la trasmissione di dati da e verso il suo esterno e in particolare da e verso il suo programmatore.
Un programma è formulato da un testo scritto in un determinato linguaggio di programmazione che in qualche modo deve essere letto e interpretato da programmatori e computers.

Nel corso della storia del calcolo automatico, dagli anni 1940 a oggi, l'organizzazione dei computer e di quanto le circonda si è evoluto continuamente in misura rilevante e in molte direzioni.
Per semplicità ci limitiamo a discutere la situazione che più qui interessa e consideriamo programma un testo scritto in un linguaggio sorgente comprensibile direttamente dai programmatori e solo indirettamente dai computers. Si deve quindi avere un processo di traduzione di ogni programma sorgente in un programma più comprensibile al computer che chiamiamo **programma eseguibile**.

B80c.05 Cominciamo a descrivere un primo schema, che denotiamo con P1, secondo il quale si ottiene la comprensione da parte del computer di un programma scritto da un programmatore e leggibile da altri.

Un tale programma, che chiamiamo programma sorgente, è costituito da stringhe di caratteri leggibili che si presentano come linee successive; ora ci limitiamo a pensare ad un programma piuttosto semplice formato da poche decine di linee, ciascuna con meno di 80 caratteri.

Questo programma sorgente viene trasformato da un programma chiamato compilatore in un cosiddetto programma compilato scritto in un cosiddetto linguaggio oggetto più comprensibile al computer del sorgente.

Su questo linguaggio diciamo solo che esprime un complesso di comandi per il computer che essenzialmente richiedono l'entrata in funzione delle istruzioni a disposizione del computer, ossia che provocano l'attivazione dei suoi circuiti operativi.

Una situazione di questo tipo si riscontra nelle usuali calcolatrici numeriche più semplici, macchine in grado di eseguire le operazioni aritmetiche e alcune altre chiamate scientifiche in grado di manipolare uno o due dati numerici espressi finitamente per ottenere un solo risultato dello stesso genere.

B80c.05 I semplici programmi che vogliamo considerare però controllano azioni più complesse, in quanto possono eseguire, oltre alle sequenze di operazioni numeriche, operazioni di lettura dall'esterno, di scrittura verso l'esterno, manipolazioni di caratteri e testi leggibili e, soprattutto decisioni basate su dati precedentemente resi disponibili che comportano di scegliere tra successive possibili sequenze di operazioni.

Inoltre si deve tener conto che il programma sorgente sottoposto al compilatore può contenere errori di tanti tipi, dai refusi ad errori di imostazione, che in qualche modo devono essere segnalati e seguiti da qualche tipo di correzione o dall'arresto della elaborazione; in ogni caso serve una segnalazione, il più possibile chiara, del tipo dell'errore e, auspicabilmente, serve qualche suggerimento di una correzione rivolto al programmatore.

Aggiungiamo che con un programma di poche decine di linee non si riescono a risolvere molti problemi; sicuramente si rendono necessari complessi di comandi e di scelte decisamente più lunghi e complessi e questo richiede attività di programmazione impegnativi che devono essere molto articolati e quindi richiedere organizzazioni ben più complesse nelle quali entrano attori che non si riducono a un programmatore e a un computer che esegue tutte le richieste che ha fornito preima di una elaborazione.

Nei paragrafi che seguono discutiamo brevemente la prima tecnica da adottare per organizzare i programmi di una certa complessità, la loro cosiddetta "organizzazione modulare"; infatti alla nozione do modulo di programma dovremo fare riferimento anche prima di presentare in dettaglio organizzazioni modulari specifiche.

In seguito dobbiamo occuparci di vari dettagli delle unità informative usate nei programmi e delle nozioni riguardanti le operazioni di immissione e di emissione di dati e solo successivamente [:g] potremo presentare i primi programmi concreti che seguono il semplice schema P1 sopra descritto.

B80c.06 Accade spesso di dover risolvere un problema impegnativo **P** con un procedimento automatico definito solo a grandi linee che si prevede richieda di eseguire molteplici manovre che coinvolgono tanti tipi di dati disponibili all'interno della organizzazione interessata, oppure ottenibili da fonti esterne attraverso la lettura di più unità periferiche e che si prevede debba affrontare diversi percorsi operativi che dipendono dalle caratteristiche delle diverse possibili istanze del problema.

In queste circostanze risulta conveniente sforzarsi di individuare problemi più ridotti relativamente semplici da risolvere che si possono considerare sottoproblemi di **P** in quanto tali che, componendo opportunamente le corrispondenti soluzioni si possa ottenere la soluzione dell'intero **P**.

Situazioni di questo genere si sono presentate spesso anche quando i problemi potevano essere risolti solo con interventi umani; in questi casi si era cercato di organizzare il lavoro di più persone che fossero

in grado di risolvere i sottoproblemi e il lavoro di un coordinatore capace di controllare il procedere dei lavori parziali e di far utilizzare le diverse soluzioni parziali fino ad assemblare un risultato finale.

Queste attività, che chiamiamo organizzazioni modularizzate, si sono riscontrate in tutti i periodi storici e in tutti i paesi, fin dalle prime civiltà dedite alla caccia e all'agricoltura.

L'organizzazione modulare si riconosce anche nella matematica, in tutti i campi della tecnica, dell'ingegneria, della medicina, nelle amministrazioni, nelle biblioteche, nei trasporti, nelle forze armate e in tutte le altre attività impegnative condotte con razionalità.

L'organizzazione modulare è stata adottata e studiata con particolare attenzione e sistematicità nelle attività computazionali basate sui dispositivi elettronici e digitali.
In questa direzione si è sviluppata la nozione di sottoprogrammi e si è giunti a disporre di collezioni di sottoprogrammi molto numerose e variegate, le cosiddette librerie di sottoprogrammi.
Questi oggetti da tempo sono considerati importanti prodotti industriali dotati di una determinante valenza economica e strategica.

B80c.07 Un programma per computer odierno, che si propone obiettivi anche solo modestamente ambiziosi presenta una struttura modulare fatta di tante componenti, i moduli di programma, in numeri che vanno dalle decine alle molte migliaia.
Vediamo che tipi di compiti svolgono i moduli che si trovano in quasi tutti i programmi con obiettivi di rilievo.

Vi sono moduli che sono incaricati di leggere dati di ingresso per ciascuna delle istanze del problema trattato.
Specularmente sono presenti sottoprogrammi che si occupano della presentazione dai risultati ottenuti da ogni esecuzione del programma.
Possono essere presenti sottoprogrammi dedicati al calcolo di valori di funzioni matematiche, dalle basilari funzioni esponenziali, logaritmiche e trigonometriche alle più complicate funzioni speciali, in conseguenza dalle possibili esigenze computazionali.
Altre esigenze applicative portano a richiedere sottoprogrammi per calcoli su matrici, sistemi di equazioni, integrali, soluzione di equazioni differenziali, calcoli di trasformate e tanto altro.
Molti sottoprogrammi disponibili riguardano le elaborazioni di ampie quantità di dati numerici per esigenze statistiche, di simulazione o di previsione.
Quando un problema richiede di servirsi dei dati disponibile in una base di dati servono i sottoprogrammi resi disponibili da un DBMS, ossia da un DataBase Management System, alcuni dei quali per l'accesso di dati, altri per la registrazione di nuovi dati nella base di dati.
I problemi riguardanti logistica, organizzazione di attività o presa di decisione possono richiedere sottoprogrammi messi a punto negli ambiti della programmazione lineare, della quadratica e della ricerca operativa.
Per molti problemi di organizzazione dei dati servono sottoprogrammi studiati nell'ambito delle teorie combinatorie; ricordiamo in particolare i programmi di ordinamento di dati numerici, di nomi, di termini o di collezioni di strutture di dati (ad esempio di dati geografici o astronomici).
Molti programmi si concludono con l'emissione di grafici, mappe e altri tipi di immagini costruite artificialmente: questi risultati richiedono sottoprogrammi dell'area della computer graphics.
Per trattare problemi riguardanti sistemi e reti di comunicazione servono sottoprogrammi riguardanti grafi.

B80c.08 Qui, senza approfondire la tipologia dei sottoprogrammi, in particolare non entrando nelle

librerie di sottoprogrammi per fini specialistici, vediamo a grandi linee come vengono strutturati i sottoprogrammi in un linguaggio procedurale come C e C++.

Innanzi tutto un sottoprogramma si può considerare come un programma con finalità circoscritte che si serve di dati di ingresso e produce risultati in uscita da e verso un unico ambiente, il programma che lo ha invocato in un certa fase della sua elaborazione.
Quindi il programma chiamante deve fare in modo di fornire i dati attuali al sottoprogramma e di ricevere i risultati ottenuti, come accade allo stesso programma, con la restrizione che sia la fornitura che la ricezione di dati vengono eseguite una sola volta per ciascun richiamo del sottoprogramma.
Come vedremo questi passaggi di informazioni si possono fare in modi diversi e quindi vi sono da distinguere diverse organizzazioni per le chiamate dei sottoprogrammi.

Risulta evidente che nei programmi molto impegnativi si devono utilizzare sottoprogrammi anch'essi piutosto impegnativi che risulta opportuno si servano di loro sottoprogrammi incaricati di lavori più semplici.
Si puoò quindi prospettare una organizzazione di un programma che si serve, direttamente o indirettamente di sottoprogrammi organizzati gerarchicamente, un modulo di tale gerarchia potendo essere richiamato da più moduli di livelli superiori diversi.
Evidentemente questi complessi di moduli di programma devono essere organizzati con grande accuratezza e seguendo criteri piuttosto elaborati che sono stati ampiamente studiati.
Tra i vantaggi di questo modo di prganizzare il software vi sono quello consistente nel riutilizzare molti sottoprogrammi per molti programmi diversi e quello di sistematizzare grandi attività di programmazione per riuscire a sviluppare una grande varietà di strumenti per la risoluzione di problemi.
Tutto questo esige la definizione di metodi organizzativi del software molto accurati e la adozione di strategie di portata molto ampia. Queste questioni vengono poste nella disciplina chiamata ingegneria del software, parte molto importante dell'informatica che costituisce anche un settorecon rilevanza economica e industriale.
L'ingegneria del software ha a che fare con tutti i maggiori linguaggi di programmazione, rispetto ai quali ha una prevedible dipendenza, ma che anche contribuisce a far evolvere per quanto riguarda lo sviluppo di nuovi linguaggi e di nuove versioni di linguaggi consolidati che devono tenere conto delle indicazioni della ingegneria del software soprattutto per quanto riguarda la produttività, la adeguabilità rispetto alla evoluzione complessiva (del software, dell'hardware e delle esigenze dei settori applicative) e della sicurezza dei sistemi computazionali.

B80 d. varietà dei sistemi hw sw

B80d.01 Nome: computer, dal francese computer dal latino com e putare = tagliare netto, rendere palese
Alternative: elaboratore elettronico, rlaboratore digitale, ordinateur calcolatore oggi usato in senso generico (regolo calcolatore o persona dedicata ai calcoli negli osservatori astronomici o capaci di efficienti calcoli mentali.
e ed è rivolto al passato

B80d.02

B80d.03

B80d.04

B80 e. informazioni booleane e numeri interi

B80e.01 Ogni dispositivo per elaborazioni automatiche e ogni macchina in grado di implementare operazioni matematiche, deve essere in grado, innanzi tutto, di trattare le entità informative più semplici ed essenziali, le cifre binarie, ossia i bits.
Questi oggetti formali servono primariamente a rappresentare i due valori di verità, il vero e il falso, valori che in genere e in particolare nel linguaggio C++ vengono associati, rispettivamente, ai numeri interi 1 e 0.
L'importanza dei bits è legata al fatto che essi consentono di controllare le scelte operative: un bit consente di scegliere tra due possibili percorsi operativi o conoscitivi. Quindi ogni meccanismo automatico dovendo essere in grado di gestire molte scelte operative, deve essere capace di operare efficientemente sui bits

In effetti le odierne tecnologie forniscono molteplici dispositivi che consentono di trattare i bits per immagazzinarli, trasformarli, ripresentarli, conservarli e trasmetterli a grandi distanze. Questi dispositivi presentano costi complessivi di realizzazione e di esercizio estremamente bassi e possono operare con velocità e tempestività molto elevate.
Inoltre oggi è possibile immagazzinare, elaborare e trasmettere grandissime quantità di informazioni binarie con dispositivi minuscoli e con costi operativi conservazione molto contenuti.

B80e.02 Come viene detto in particolare in B60, ai valori di verità si possono applicare le operazioni booleane concernenti le sentenze, ossia gli enunciati ai quali si è convenuto possibile assegnare uno di tali valori.
Un bit, considerato portatore di un valore di verità, in date circostanze, ovvero dopo aver fissate opportune convenzioni, fornisce l'informazione che determina una scelta dicotomica, cioè una scelta tra due alternative mutuamente esclusive.
Una tale scelta in genere riguarda la effettuazione o meno di una data azione, oppure l'esecuzione di una di due manovre diverse; gli effetti ottenibili servendosi di queste celte basilari rende i valori di verità fondamentali nello studio r nelle applicazioni delle procedure.

B80e.03 Un bit può essere usato anche per esprimere la presenza o l'assenza di un particolare oggetto in una prefissata collezione. Per trattare questioni concernenti l'appartenenza o meno ad un insieme finito E presentato con una sequenza, ovvero con una lista, di n componenti risulta naturale utile servirsi di sequenze di n cifre binarie: infatti ciascuna di tali sequenze consente di individuare un sottoinsieme di E, in quanto di tale insieme costituisce la funzione indicatrice [B13b].
Le sequenze binarie sono quindi assai utili per il controllo delle collezioni di dati (in particolare nell'ambito dei `DBMS (we)`, i sistemi per la gestione delle basidati).

Mediante sequenze di bits si possono rappresentare numeri interi, [B12c. Osserviamo che anche questa prestazione si può ricondurre ad indicazioni di presenze oppure assenze di oggetti: precisamente per esprimere i numeri naturali costituenti l'intervallo $\mathbf{[}0 : 2^h - 1\mathbf{]}$, gli oggetti che possono essere presenti o meno sono le potenze 2^i nelle espressioni numeriche $\sum_{i=0}^{h} b_i\, 2^i$ nelle quali $b_i \in \{0, 1\}$.

Bisogna aggiungere due altri ruoli fondmentali dei bits.
Tutti i circuiti digitali sono costituiti da componenti che eseguono una delle operazioni binarie: gli operandi o l'unico operando di una di queste operazioni sono forniti da impulsi elettrici chiaramente

attribuibili a due classi, quella rappresentante il bit 0 e quella corrispondente al bit 1; un impulso simile rappresenta il risultato dell'operazione.
Le memorie a stato solido, attualmente di largo uso, sono costituite da microcircuiti NAND in grado di conservare e restituire segnali di due tipi, ciascuno dei quali atto a rappresentare un valore binario.

B80e.04 La tecnologia odierna consente di registrare, trasmettere ed elaborare con grande efficienza e velocità enormi quantità di cifre binarie.
Sono disponibili dischi magnetici e banchi di memorie a stato solido in grado di registrare molti triliardi di bits, ovvero molti multipli di 10^{12} di bits e banchi di dispositivi circuitali statici (memorie a stato solido) con capacità paragonabili.
I circuiti operativi dei processori attuali sono in grado di elaborare informazioni binarie a velocità di miliardi di operazioni al secondo.
Come si ricava da una serie di situazioni, tutte le informazioni trattabili con le attuali apparecchiature informatiche possono essere rappresentate mediante sequenze binarie.

Le precedenti considerazioni rendono ben comprensibile la attuale tendenza generale di servirsi di sequenze di bits per la registrazione, la trasmissione e l'elaborazione di tutte le informazioni che si vogliono gestire mediante automatismi (dati numerici, testi, firme, immagini, suoni, animazioni, programmi, catene dimostrative, ...).

Procediamo ora a illustrare le rappresentazioni binarie dei tipi più semplici di informazioni da trattare con il computer e da gestire con linguaggi di programmazione.

B80e.05 Per rappresentare le informazioni delle diverse specie si rende necessario che i dispositivi hardware e i linguaggi di programmazione consentano di controllare unità di registrazione di diverse capacità.
Occupiamoci per ora dei numeri interi.

Mentre per esporre proprietà matematiche generali degli interi in genere non ci si deve preoccupare di quanto siano grandi, per decidere come implementarli si deve essere consapevoli che si possono trattare con semplicità e uniformità solo interi che variano in intervalli limitati.

Si osserva che si possono trattare con maggiore efficienza interi che possono assumere solo valori assoluti limitati, mentre si possono trattare con maggiore versatilità interi che possono assumere valori massimi maggiori, ma che necessariamente ciascuno di essi richiede più bits richiede dispositivi di memoria più costosi e circuiti operativi più estesi e che richiedono più energia o maggiori tempi esecutivi.

In talune circostanze conviene operare con una collezione di numeri di valori limitati ma che si possono elaborare efficientemente; in altre è preferibile una collezione numerica più estesa che si possa trattare con meno preoccupazioni sui massimi valori che possono assumere, ma più costosa in termini di dispositivi da impiegare, di tempi esecutivi, di energia consumata e di riscaldamento da tenere contenuto.

Prevalentemente si trattano interi rappresentabili con 16, 32, 64 o 128 bits e si può stabilire se si vogliono solo numeri nonnegativi oppure numeri che possono essere sia nonnegativi che negativi.

Con unità di registrazione da 16 bits si possono trattare gli interi naturali facenti parte dell'intervallo $\mathbf{[}0 : 65\,535\mathbf{]}$ ($2^{16} = 65\,536$), oppure i numeri interi relativi [B20a, `Complemento a due (wi)`) che possono assumere i valori nell'intervallo $\mathbf{[}-32\,768 : 32\,767\mathbf{]}$.

Con sequenze di 32 bits si possono trattare gli interi naturali dell'intervallo $\mathbf{[}0 : 4\,294\,967\,295\mathbf{]}$ ($2^{32} = 4\,294\,967\,295$), oppure gli interi relativi appartenenti a $\mathbf{[}-2\,147\,483\,648 : 2\,147\,483\,647\mathbf{]}$.

Con sequenze di 64 bits, dato che $2^{64} = 18\,446\,744\,073\,709\,551\,616$, si possono trattare gli interi naturali dell'intervallo $\mathbf{[}0 : 18\,446\,744\,073\,709\,551\,616\mathbf{]}$, oppure gli interi relativi dell'intervallo $\mathbf{[} -9\,223\,372\,036\,854\,775\,808 : 9\,223\,372\,036\,854\,775\,807\mathbf{]}$.

B80e.06 Nel seguito faremo riferimento quasi esclusivamente a personal computers di uso comune, le macchine più facilmente disponibili per sperimentare, ma gran parte delle considerazioni che seguono possono essere applicate ai molti altri tipi di apparecchiature per la gestione di informazioni digitali (tablets, smart phones, fotocamere, dispositivi telematici, smart TV, rilevatori, servomeccanismi, apparecchiature medicali, ...).

Inizialmente ci occuperemo delle funzioni per la registrazione delle informazioni digitali, ovvero vedremo i computers e le altre apparecchiature con intelligenza digitale come contenitori di grandi sequenze di bits.

I dispositivi per la registrazione delle informazioni binarie, sostanzialmente i circuiti elettronici degli odierni computers, li chiamiamo **dispositivi di memoria**.
I dispositivi, che consentono di trattare (immettere, trasformare ed emettere) le informazioni digitali per la massima parte riguardano sottosequenze di lunghezza ben definita di registri binari con ben definiti insiemi di valori.
Quindi per molte questioni i dispositivi di memoria si possono convenientemente considerare costituiti da sequenze di queste sottosequenze.
Queste sottosequenze binarie dal punto di vista dei linguaggi di programmazione le chiameremo **celle indirizzabili di memoria**, in breve **celle-mm**.

B80e.07 Nelle odierne apparecchiature elettroniche vengono trattate soprattutto celle-mm riguardanti sottosequenze di 8, 16, 32, 64 e 128 bits. Le celle-mm di s registri binari le diremo in breve celle di s bits o anche celle-sb.

Chiameremo **bytes** o **ottetti** i contenuti delle celle-mm di 8 bits.), **halfwords** o semiparole le celle di 16 bits, **words** o parole quelle di 32 bits, **doublewords** o doppie parole quelle di 64 bits e **quadwords** le sottosequenze di 128 registri binari.

Per molte considerazioni le posizioni binarie di ciascuna delle celle-sb conviene raffigurarle come caselle quadrate, boxes, disposte orizzontalmente e caratterizzando le loro posizioni con gli interi 0, 1, 2, ..., $s-1$ crescenti quando si procede da sinistra verso destra.
Queste caselle hanno il ruolo delle celle binarie, ovvero sono destinate a contenere valori binari che possono essere modificati nel corso di una elaborazione e gli interi che forniscono le loro posizioni si possono chiamare i loro indirizzi interni.

Abbiamo quindi raffigurazioni come le seguenti:

//input pB70b05

B80e.08 Nelle considerazioni che seguono presentiamo le risorse di memoria seguendo lo schema che viene proposto ai programmatori in un linguaggio di alto livello come C e C++.
Questo schema costituisce una semplificazione standardizzata di quanto si riscontra fisicamente, ossia al livello operativo dei circuiti hardware dei computres attuali. Le memorie presentate ai programmatori sono vicine alle memorie materiali di cui erano dotati i computers strettamente sequenziali degli anni 1960-1970, nei quali si aveva un unico tipo di memoria centrale.
I sistemi successivi, oltre a differenziarsi notevolmente, hanno adottato tecniche via via più articolate e complesse volte ad aumentare le prestazioni complessive utilizzando i dispositivi hardware dei diversi generi che si rendevano disponibili, dispositivi a diversi livelli che in genere di dispongono in successivi

strati dai più vicini ai circuiti operativi ai più lontani, strati nei quqli si hanno velocità operative decrescenti, costi unitari decrescenti ed estensioni crescenti; con questo genere di organizzazione si devono adottare strategie organizzative e procedure di supporto necessariamente più elaborate.

Le schema semplificato che seguiremo ha due importanti pregi: consente ai i programmatori di produrre programmi che si adattano a una grande varietà di sistemi harware e mantengano la loro efficacia nel tempo; evita che i programmatori si debbano occupare dei complesssi meccanismi dietro le quinte sopra accennati.
delle memorie (virtuali) su dischi e banchi a stato solido.

B80e.09 Secondo lo schema delle memorie semplifcato per il programmatore, il programma dispone di una memoria centrale nella quale si trovano le celle-mm di diverse estensioni.
Ciascuna di esse può essere raggiunta dai circuiti operativi a della unit a di controllo per essere letta o scritta attraverso il suo **indirizzo**, l'intero naturale che rappresenta la sua posizione nella sequenza (virtuale) che costituisce la memoria centrale.

Il programmatore non deve occuparsi della disposizione nella memoria centrale, di questa si incarica il sistema di gestione dei programmi.
Il programmatore gestisce singole celle-mm, sequenze di celle-mm o schieramenti di celle-mm di due o più dimensioni riconducibili a sequenze attraverso i loro identificatori la cui scelta è a sua completa disposizione.
In tal modo egli riesce a controllare pienamente le informazioni che lo interessano e nient'altro; l'unica sua preoccupazione riguarda la estensione complessiva di memoria centrale a sua disposizione.

Occorre anche segnalare che tra i diversi ruoli delle celle-mm vi è anche quello di individuare le posizioni di altre celle-mm; l'estensione delle celle-mm dedicate agli indirizzi evidentemente dipende dalla ampiezza della memoria disponibile, una memoria più ampia richiede indirizzi più lunghi.

B80e.10 Coerentemente con quanto visto per le celle elementai binarie di una cella-mm, conviene raffigurare le sequenze di celle-mm con file di rettangoli disposti orizzontalmente caratterizzati da indirizzi che crescono procedendo da sinistra verso destra.
Può servire visulizzare il contenuto di una sequenza di celle-mm indicando in questi rettangoli i valori dei rispettivi contenuti correnti e segnalando qualche indirizzo . Il valore variabile nel tempo del contenuto di una cella-mm può essere rappresentato diversamente a seconda del suo ruolo, ovvero dell'utilizzo al quale è destinato, o a seconda del significato che gli si attribuisce in una particolare presentazione di alcune clle-mm.
Torneremo sull'argomento parlando di variabili e operazioni relative richieste nel linguaggio.

Occorre precisare che vanno distinti gli indirizzi assegnati alle diverse taglie di celle-mm, cioè ai bytes, alle halfwords, alle words etc.
Va anche segnalato che l'estensione della memoria di un computer o di altri dispositivi digitali di solito viene espressa mediante il numero dei suoi bytes.
Le capacità delle memorie più comodamente utilizzabili da un computer attuale può superare il terabyte.

B80e.11 Per trattare gli indirizzi e i contenuti della memoria centrale conviene avere presenti le potenze di 2 [W10a01].
Vediamo un esempio abbastanza realistico di una memoria di $2^{30} = 1\,073\,741\,824$ bytes, cioè di $2^{33} = 8\,589\,934\,592$ bits; per la misura di queste grandezze si usano notazioni come 1 GB e 8 GB, anche se notazioni per maggiore precisione si dovrebbero usare le notazioni 1 Gibiby e 8 Gibiby.

La nostra memoria può anche essere vista come sequenza di $2^{29} = 536\,870\,912$ halfwords, come sequenza di $2^{28} = 268\,435\,456$ words, come sequenza di $2^{27} = 134\,217\,728$ doublewords e come sequenza di $2^{26} = 67\,108\,864$ quadwords.
Si può quindi indirizzare un byte in memoria con un intero che, idealmente, va da 0 a 1 073 741 823, una halfword con un intero compreso tra 0 e 536 870 911, una word con un intero da 0 a 134 217 727, una doubleword con un intero appartenente a $[0 : 134\,217\,728]$ ed una quadword con un intero naturale minore o uguale a 67 108 863.
I bytes individuabili nella memoria centrale occupano ottetti la cui prima posizione binaria è un multiplo di 8, le halfwords sequenze la cui prima posizione è un multiplo di 16 e così via.
In una sequenza di 128 bits della memoria centrale il cui primo bit occupa la posizione `i` (intero multiplo di 128, la lunghezza delle doublewords) si possono vedere 16 bytes, oppure 8 halfwords, oppure 4 words, oppure due doublewords oppure una quadword, come dal seguente schema.

//input pB70b08

B80e.12 Come si è detto, i contenuti di ciascuna delle celle-mm possono rappresentare oggetti diversi in dipendenza del ruolo che alla cella il programmatore sta attribuendo.
Il ruolo di una cella-mm viene stabilito a livello software da una dichiarazione nel linguaggio di programmazione, mentre a livello hardware viene preso in considerazione dai circuiti operativi che operano sulla cella.

Una cella-mm di s bits può rappresentare un intero naturale compreso tra 0 e $2^s - 1$. Per esempio la halfword che contiene la sequenza di bits $b_0 b_1 b_2 \cdots b_{15}$ può rappresentare l'intero $\sum_{i=0}^{15} b_i\, 2^i$.

Gli interi dell'intervallo $[0 : 2^s - 1]$ sono detti **unsigned integers** su s bits.

Conviene soffermarsi a osservare alcuni interi naturali rappresentati dalle halfwords. Lo zero è rappresentato da 00000000 00000000, i successivi 5 interi da
10000000 00000000, 01000000 00000000, 11000000 00000000, 00100000 0000000 e 10100000 00000000.
Gli interi 10, 100, 1000 e 10000 sono forniti, rispettivamente, dalle sequenze
01010000 00000000, 00100110 00000000, 00010111 11000000 e 00001000 11110100.
Il massimo degli unsigned integers su 16 bits, $2^{16} - 1 = 65\,535$, è rappresentato da 1111111 11111111.

B80e.13 Una cella-mm di s bits la chiameremo **cella-*s*b** può rappresentare anche una sequenza di s valori booleani, valori da interpretare come falso se il bit vale 0 e come vero se il bit vale 1.

Ricordando la nozione di funzione caratteristica di un sottoinsieme entro un insieme finito, possiamo anche affermare che una cella-sb consente di individuare un sottoinsieme di un insieme ambiente di al più s elementi che sia stato dotato di un ordine.
Per esempio il sottoinsieme dell'insieme dei mesi dell'anno i cui nomi italiani contengono la lettera "r" è esprimibile con la halfword $011100001111_2 = 3854_{10}$.

B80e.14 L'insieme (finito) di interi che è più utile rappresentare con s bits è l'intervallo $[-2^{s-1} : 2^{s-1} - 1$. In particolare le celle-16b possono rappresentare gli interi dell'intervallo $] = [-32736 : +32765]$, le celle-32b gli interi compresi tra $-2\,147\,483\,648$ e $2\,147\,483\,647$, le celle-64b gli interi da $-9\,223\,372\,036\,854\,875\,808$ a $9\,223\,372\,036\,854\,875\,807$.

Questi interi li chiameremo **signed integers** e la loro adozione, in confronto con quella degli unsigned integers, rinuncia a una metà di questi interi nonnegativi, ma consente di disporre di un ugual numero, 2^{s-1}, di numeri negativi.

Questi sono determinati secondo la cosiddetta **rappresentazione in complemento a 2**. Secondo essa i numeri nonnegativi sono caratterizzati da 0 nella posizione $s-1$ e la sequenza degli $s-1$ bits precedenti $b_0b_1b_2\cdots b_{s-2}$ fornisce l'intero $\sum_{i=0}^{s-2} b_i\,2^i$.
Il generico intero negativo è dato dalla sequenza della forma $b_0b_1b_2\cdots b_{s-2}1$ ed il suo valore si ottiene dalla sequenza dei bits complementari $\overline{b}_0\overline{b}_1\overline{b}_2\cdots\overline{b}_{s-2}0$ con $\overline{b}_i := 1-b_i$ e togliendo 1 dal valore di questa sequenza considerate unsigned integer; il valore è quindi $-\sum_{i=0}^{s-2}\overline{b}_i\,2^i-1$.

Vediamo le rappresentazioni di alcuni interi negativi mediante halfwords:

$$-32\,766=-2^{15} \text{ dato da } 00000000\,00000001 \quad , \quad -32\,765 \text{ dato da } 10000000\,00000001 \quad ,$$
$$-100 \text{ dato da } 00111001\,11111111 \quad , \quad -16 \text{ dato da } 00001111\,11111111 \quad ,$$
$$-7 \text{ dato da } 11101111\,11111111 \quad , \quad -1 \text{ rappresentato da } 11111111\,11111111 \quad .$$

B80e.15 Può servire osservare che se n denota un signed integer dato dalla sequenza binaria $b_0b_1\cdots b_{s-1}$ ed $\overline{n}$ il suo complemento a uno, cioè il numero espresso da $\overline{b}_0\overline{b}_1\cdots\overline{b}_{s-1}$, la loro somma fornisce la sequenza di s bits uguali ad 1, cioè $n+\overline{n}=-1$; quindi vale la formula

$$\overline{n}=-n-1 \quad .$$

Osserviamo che per la rappresentazione in complemento a 2 la sequenza crescente delle stringhe binarie dopo lo 0 vede i successivi numeri positivi e dopo l'ultimo, $2^{s-1}-1$, vede i numeri negativi sempre in ordine crescente a partire dal minimo, -2^{s-1} fino a -1; questo è quello che in ordine ciclico precede lo 0.

Questa rappresentazione, a prima vista piuttosto bizzarra, in pratica offre precisi vantaggi per la realizzazione dei circuiti operativi che implementano le operazioni aritmetiche e le relazioni d'ordine e viene adottata da tutti i costruttori di apparecchiature digitali costituendo un importante standard internazionale de facto.

Segnaliamo inoltre che le words (sequenze di 32 bits) consentono di trattare gli interi-mm dell'intervallo $[-2\,147\,483\,648 : 2\,147\,483\,647]$, che l'insieme degli interi-mm contenuti nelle doublewords (64 bits) è $[-9\,223\,372\,036\,854\,875\,808 : 9\,223\,372\,036\,854\,875\,807]$ e che le quadwords consentono di registrare ciascuno degli interi compresi tra $-170\,141\,183\,145\,469\,231\,731\,687\,303\,715,884\,105\,728$ e $-170\,141\,183\,145\,469\,231\,731\,687\,303\,715,884\,105\,727$.

B80e.16 Per taluni scopi risulta opportuno considerare anche celle di memoria corrispondenti a quaterne di bits; a queste celle-mm si da il nome di **nibbles**, ossia di bocconi.
I possibili valori di un nibble sono evidentemente 16 e sono in corrispondenza biunivoca con un intero dell'intervallo $[0:15]$; essi possono essere posti in biiezione con una cosiddetta **cifra esadecimale**.
Per queste entità vale la seguente tavola di conversione

0000	1000	0100	1100	0010	1010	0110	1110	0001	1001	0101	1101	0011	1011	0111	1111
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Per esprimere i contenuti delle unità di memoria può essere vantaggioso servirsi delle notazioni esadecimali, in base 16.
Chiaramente un byte equivale a 2 nibbles, una halfword a 4 nibbles, una word ad 8 nibbles e così via. Quindi si può esprimere il contenuto di un byte con una coppia di cifre esadecimali, il contenuto di una halfword con una quaterna di cifre esadecimeli, il contenuto di una words con 8 cifre esadecimali e così via.

Le notazioni esadecimali sono spesso utilizzate per esprimere i contenuti attuali di ampie zone della memoria centrale e per esprimere gli indirizzi della memoria centrale. Per esempio se si vogliono individuare gli indirizzi dei bytes di una memoria di un gibibyte, cioè di una memoria di $2^{30} = 1\,073\,741\,824$ bytes servono 15 cifre esadecimali: il primo byte corrisponde all'indirizzo 000 000 000 000 000, l'ultimo all'indirizzo $FFF\ FFF\ FFF\ FFF\ FFF$.

B80e.17 Stante la finitezza delle risorse di memoria disponibili con un computer, in ogni programma ci si deve preoccupare di non superare i limiti che derivano dalla finitezza del computer in uso.
Accenniamo alle precauzioni da assumere per elaborazioni sui numeri interi.

Se si devono trattare interi piccoli si possono usare per essi delle halfwords; se si può essere sicuri che non si incontrano interi al di sopra del miliardo si possono usare le words; altrimenti bisogna servirsi di doublewords o di quadwords.
Per trattare interi di grandezza maggiore oppure interi di grandezza imprevedibile servono sottoprogrammi che operano su sequenze di interi dei tipi precedenti e che sono in grado di rappresentare valori interi molto elevati attraverso sequenze di celle-mm della estensione opportuna che potrebbe essere prevedibile solo nel corso della costruzione dei suddetti nuovi valori.
Evidentemente più si vuole essere sicuri di non superare i limiti delle memorie, più si devono impegnare risorse di memoria, di tempo di esecuzione e di onere di programmazione.

Va anche detto in linea di massima che le grandi prestazioni dei computers attualmente disponibili consigliano di scrivere programmi che non si curano di risparmiare le risorse, ma che abbiano elevata adattabilità, ovvero elevata affidabilità e in particolare che non incorrano in errori di superamento dei limiti per i numeri interi incontrati.
Va inoltre segnalato che i sistemi di programmazione disponibili sono in grado di segnalare molti degli inconvenienti accennati.
Un comportamento spesso consigliabile consiste nello scrivere programmi provvisori semplici, anche se rischiosi, di provarli attentamente e se giudicato necessario, di redigere programmi più definitivi ben accurati rispetto alle prestazioni che possono portare a risultati condizionati da superamento dei valori massimi garantiti.

Segnaliamo anche un altro possibile comportamento di programmazione. Di fronte a problemi impegnativi e poco chiari può essere opportuno, grazie al fatto che il costo delle sperimentazioni con il computer in linea di massima è piuttosto basso, si possono effettuare tentativi con programmi poco approfonditi al fine di chiarire gradualmente le caratteristiche del problema e di una procedura per risolverlo con l'osservazione di un certo numero di risultati empirici preliminari.

Alle precedenti considerazioni sopra gli insiemi finiti di interi trattabili si dovranno aggiungere considerazioni riguardanti quelli che chiamiamo real numbers; queste altre considerazioni oltre a riguardare i valori massimi di questi numeri, concernono i loro possibili valori assoluti minimi (zero escluso) e la loro precisione; prevedibilmente saranno considerazioni un poco più complesse.

B80 f. informazioni simboliche

B80f.01 Vediamo ora come vengono trattate comunemente nel linguaggio C++ i caratteri leggibili e le **informazioni simboliche**, in pratica le stringhe di caratteri appartenenti a un preciso alfabeto.
Questo alfabeto comprende le cifre decimali, le lettere dell'alfabeto inglese, 26 maiuscole e 26 minuscole, segni di interpunzione e alcuni segni che denotano operazioni matematiche.
Oltre a questi segni visualizzabili sono trattabili alcuni caratteri che svolgono ruoli importanti nel controllo dei dispositivi periferici come stampanti e schermi video e nel controllo della trasmissione delle informazioni con altri computers e con apparecchiature digitali che seguono precise regole operative concordate.

le stringhe dei caratteri trattabili consentono di elaborare gli usuali testi leggibili che possono essere stampati, presentati su uno schermo, registrati su un dispositivo di memoria portatile (disco o chiavetta flash), o emessi da un altoparlante e che possono essere immessi nel sistema digitati su tastiera, scritti a mano su uno schermo adeguato, catturati da un microfono o presi da un dispositivo di memoria trasferibile.

Con il linguaggio C++ e con molti altri linguaggi, i dati simbolici si trattano seguendo soprattutto il sistema di codifica detto **codice ASCII** memorizzando ogni stringa in una sequenze di bytes consecutivi, un carattere per byte.

Altre modalità consentono di eleborare alfabeti più ampi con tecniche più elaborate; tra queste sono prevalenti quelle del cosiddetto **codice Unicode**, sistema di codifica che si è proposto di rendere trattabili con modalità standard la massima parte delle lingue naturali e artificiali effettivamente praticate nel presente e nel passato che presentiamo in :d04-d05.

B80f.02 ASCII è l'acronimo di American Standard Code for Information Interchange e costituisce uno standard internazionale ampiamente riconosciuto.
Per la precisione occorre distinguere tra le due varianti ASCII-7 ed ASCII-8 che registrano un carattere, rispettivamente con 7 e con 8 bits.
La prima variante è stata ampiamente utilizzata a partire dagli anni 1965-1970, periodo nel quale si sono imposti i sistemi in grado di gestire le celle-mm di 8, 16, 32, 64 e 128 bits.
La seconda è un ampliamento della prima e include un buon numero di caratteri visualizzabili delle maggiori lingue occidentali e caratteri utilizzati per la più semplice grafica e per i più semplici videogiochi.

ASCII dedica i primi 32 bytes, corrispondenti ai valori decimali da 0 a 31, ai valori binari da 0000000_2 a 00011111_2 e ai valori esadecimali da 00_{16} a $1F_{16}$ e il byte $127 = 01111111_2 = 7f_{16}$ ai cosiddetti **caratteri di controllo**.

0	00	NUL	^	\0	null	16	10	DLE	^P	\r	data link escape
1	01	SOH	^A		start of heading	17	11	DC1	^Q		device control 1
2	02	STX	^B		start of text	18	12	DC2	^R		device control 2
3	03	ETX	^C		end of text	19	13	DC3	^S		device control 3
4	04	EOT	^D		end of transmission	20	14	DC4	^T		device control 4
5	05	ENQ	^E		enquiry	21	15	NAK	^U		negative acknowledgement
6	06	ACK	^F		acknowledgement	22	16	SYN	^V		synchronous idle
7	07	BEL	^G	\a	bell	23	17	ETB	^W		end of transmission block

8	08	BS	^H	\b	backspace	24	18	CAN	^X		cancel
9	09	HT	^I	\t	horizontal tab	25	19	EM	^Y		end of medium
10	0A	LF	^J	\n	line feed	26	1A	SUB	^Z		substitute
11	0B	VT	^K	\v	vertical tab	27	1B	ESC	^[		escape
12	0C	FF	^L	\f	form feed	28	1C	FS	^\		file separator
13	0D	CR	^M	\r	carriage return	29	1D	GS	^]		group separator
14	0E	SO	^N		shift out	30	1E	RS	^^		record separator
15	0F	SI	^O		shift in	31	1F	US	^_		unit separator
127	7F	DEL	^?		delete						

B80f.03 I 95 bytes con valori numerici compresi tra 20_{16} e $7E_{16}$ sono dedicati ai caratteri visualizzabili o stampabili, cioè alle 26 lettere dell'alfabeto romano-inglese, alle cifre decimali, ai segni di interpunzione e alcuni segni per operazioni matematiche.

La tabella che segue presenta le loro codifiche.

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
(spazio)	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	
p	q	r	s	t	u	v	w	x	y	z	{	|	}		

B80f.04 Tra i produttori di hardware intorno al 1970 ha cominciato ad imporsi l'organizzazione che favorisce l'indirizzamento di unità di 8 bits chiamate **bytes** o ottetti, unità i cui valori si possono rappresentare con interi da 0 a 255 o con coppie di cifre esadecimali (da 00_{16} a ff_{16}).
Per ogni valore binario si tende a utilizzare un intero byte, per gli interi da 16 bits due bytes consecutivi, per gli interi da 32 bits si utilizzano 4 bytes consecutivi, per gli interi da 64 bits 8 bytes consecutivi e per 18 bita 16 bytes consecutivi.

L'importanza dei bytes è dovuta all'ampio utilizzo di collezioni di caratteri con un numero di segni tra una e due centinaia. ASCII. `ASCII (we)`, acronimo di American Standard Code for Information Interchange, denota un sistema di codifica diffusosi ampiamente nella elaborazione automatica dei dati negli anni 1965-1970. Esso si serve di 7 bits per rappresentare 128 caratteri con i quali in quegli anni si riusciva a soddisfare gran parte delle esigenze delle elaborazioni simboliche. Il relativo alfabeto comprende 26+26 lettere minuscole e maiuscole, cifre decimali, coppie di parentesi ("`(`", "`)`", "`[`", "`]`", "`{`", "`}`"), segni di interpunzione ("`,`", "`.`", "`:`", "`;`", "`?`", "`!`", "`"`") i segni matematici di largo uso ("`+`", "`-`", "`=`", "`<`", "`>`", "`^`"), pochi altri caratteri visualizzabili ("`#`", "`&`", "`|`", "`*`", "`$`", "94", "`'`", "`_`", "\", ...) e 32 segni utilizzati per le telecomunicazioni.

Successivamente si è avuto un primo ampliamento dei caratteri facilmente trattabili con i codici che genericamente sono richiamati dal termine `ASCII esteso (wi)`; questi mediante ottetti di bits consentono di rappresentare 256 caratteri.
Esistono diverse varianti di ASCII esteso che differiscono per una parte dei 128 caratteri non ASCII originali in relazione alle esigenze delle lingue di paesi, sostanzialmente occidentali, nei quali sono utilizzate o di alcune apparecchiature specifiche.

Qui faremo riferimento soprattutto alla codifica chiamata Latin-9 utilizzata nell'Europa Occidentale e standardizzata dall'ISO con il nome ufficiale `ISO 8859 (we)`-15.
Secondo questi sistemi le 26 lettere dell'alfabeto romano-inglese maiuscolo sono codificate con gli ottetti contraddistinti dai valori da 65 a 90, le minuscole con i valori da 97 a 122 e le cifre decimali con i valori da 48 a 57.

B80f.05 Il termine **Unicode** denota un importante e ambizioso progetto internazionale avviato nel 1991 che si è proposto di standardizzare l'implementazione digitale della maggior parte dei caratteri e degli altri segni utilizzati nei testi prodotti in tutte le parti del mondo, nel presente e nel passato, comprendendo testi letterari, matematici, musicali e testi dedicati alle svariate tecniche.

Il suo scopo è quello di agevolare il più possibile la circolazione digitale e telematica dei dati, delle informazioni, delle conoscenze e conseguentemente delle idee.

Il progetto Unicode viene portato avanti dal consorzio Unicode Consortium sostenuto dalle importanti organizzazioni ISO e IEC e si occupa della codifica digitale dei caratteri e dei segni che organizza e classifica nel cosiddetto **Universal Coded Character Set**, abbreviato da **UCS**.

Nel suddetto character set sono contenuti tutti gli alfabeti dei linguaggi naturali del presente e del passato della cui utilizzazione, scritta od orale, si riesce a trovare una accettabile documentazione.
In UCS vengono collocati anche i caratteri e i segni usati in una ampia gamma di linguaggi artificiali e di sistemi di scrittura.
Nella versione di Unicode 16.0.0 del 2024-09-10 risultano definiti 154 998 caratteri suddivisi in blocchi di simboli [https://en.wikipedia.org/wiki/Unicode_symbol] e 168 cosiddetti "scripts", collezioni di segni utilizzati negli scritti prodotti da attività comunicative, accademiche, industriali e tecniche in senso lato [https://www.unicode.org/charts/PDF/Unicode-16.0/U160-1CC00.pdf].

Esempi di blocchi di caratteri: simboli delle divise monetarie, segni a deponente e a esponente, numeri enumeri Maya, lettere incorniciati in cerchi, simboli dei sistemi fonemici e fonetici (IPA), emoji ed emoticon, frecce, operatori matematici, forme geometriche, simboli pittografici, simboli musicali, simboli per carte da gioco, scacchi, domino, mahjong, ...

Esempi di scripts : geroglifi egizi, simboli pittografici, simboli di sistemi di scrittura sillabici, simboli delle lingue asiatiche, africane e di tutte le altre lingue esotiche, simboli utilizzati da prodotti informatici

anche tra quelli ora in disuso, simboli per circuiti elettrici ed elettronici, frecce, simboli stradali, simboli alchemici ed ermetici, ...

B80f.06 Il sistema di codifiche Unicode è uno standard internazionale de facto e trae grande importanza dal fatto di dare delle regole ampiamente condivise per la circolazione su Internet di un flusso di messaggi ormai gigantesco e in continua crescita.
In effetti Unicode viene supportato da un gran numero di prodotti software di vasta adozione: linguaggi di programmazione, compilatori, IDE, linguaggi per la comunicazione come HTML, browsers, piattaforme per posta elettronica, DBMS, sistemi operativi, traduttori automatici, generatori di immagini e suoni, ...

Questo sistema di codifiche mette a disposizione 16, 20 o 24 bits per ogni codepoint, sequenza di bits che costituisce la codifica di un carattere o script facente parte di UCS secondo uno schema un po' elaborato.
Lo schema prevede 17 cosiddetti planes, ciascuno costituito da $65\,536 = 2^{16}$ codepoints contigui e quindi consente di trattare fino a 1 114 112 simboli.
Per ottimizzare statisticamente memorizzazione, trasmissione e eleborazione dei testi si assegnano codepoints più corti ai segni per i quali si prevede un utilizzo più frequente.

B80 w. prima vista d'assieme

B80w.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.

B80w.02 Allo sviluppo e all'utilizzo di un programma impegnativo possono lavorare parecchie persone. Ora però consideriamo un solo tipo di operatore umano, il programmatore, la persona incaricata di definire nei dettagli la soluzione di un problema e a tradurla in un programma scritto in un determinato linguaggio.

Più avanti per descrivere i possibili vari aspetti dello sviluppo dei programmi prenderemo in esame altri tipi di figure professionali e accenneremo anche alla possibilità di far intervenire procedure automatiche per sostenere o eseguire interamente alcune fasi del cosiddetto "ciclo di vita del software; queste procedure sono attribuibili all'area della AI, dell'intelligenza artificiale.

Un programma nasce quando viene scritto il **suo testo sorgente**, testo scritto in uno o più files normalmente leggibili, seguendo le regole formali proprie del linguaggio.
Il testo sorgente di un programma, chiamato anche **codice sorgente** deve contenere il complesso delle richieste rivolte ad un computer che consentano a questa macchina di affrontare con le sue sole capacità operative, un problema in una certa gamma ben definita delle sue "istanze", cioè delle situazioni specifiche nelle quali si può presentare.

Un testo sorgente deve essere tradotto dal linguaggio di programmazione, C++, in un linguaggio comprensibile dalla macchina.
La traduzione viene effettuata da altri programmi, i compilatori o gli interpretatori, i quali evidentemente devono essere in grado di conoscere sia il linguaggio C++ che il linguaggio del computer.

I programmi devono essere considerati prodotti industriali di un genere particolare: si parla di "prodotti software".
Quindi l'organizzazione complessiva delle attività richieste dai programmi è sempre stata fortemente influenzata dai molteplici interessi che ruotano intorno alle elaborazioni automatiche e dallo stato, in

continua evoluzione degli efficaci strumenti che vengono via via resi disponibili per sostenere lo sviluppo delle attività riguardanti l'elaborazione automatica dei dati e in primis i programmi stessi.

B80w.03 Vediamo a grandi linee come viene eseguita la traduzione dei programmi.

Cominciamo con il caso di un programma molto semplice, ad esempio un programma didattico che serve a introdurre un numero ridotto di elementi del linguaggio che potrebbe portare anche ad una sola esecuzione. Alcuni di questi programmi si incontrano nelle pagine che seguono.

Il suo testo sorgente consiste in poche decine di righe che richiedono di leggere alcuni dati di partenza, l'esecuzione di poche operazioni che coinvolgono le informazioni lette e poche altre disponibili all'interno del programma e la scrittura per l'esterno delle informazioni costituenti i risultati della esecuzione del programma.
La traduzione di un tale programma potrebbe essere eseguita da un compatto programma traduttore e subito seguita dalla esecuzione delle manovre programmate.

In generale si hanno programmi di media complessità che richiedono manovre piuttosto articolate che potrebbero essere eseguite varie volte, in vari contesti applicativi, con dati di ingresso piuttosto dissimili nei diversi contesti, servendosi di computers diversi e con emissioni di risultati complesse, ad esempio portate avanti in tempi successivi pe il successivo presentarsi di richieste diverse.
A questi programmi nel seguito faremo riferimento con due finalità: spiegare i motivi di molte scelte riguardanti il linguaggio C++ e gli strumenti che lo accompagnano e motivare i consigli che vanno rivolti ai programmatori affinché il loro lavoro possa portare a programmi che possono considerarsi prodotti validi secondo uno o più dei punti di vista dai quali secondo i quali possono venire valutati.

Si possono poi immaginare programmi con importanti compiti che possono richiedere testi sorgente di milioni di linee scritte, molte decine di programmatori e di esperti al loro fianco, richiedere dati di molteplici provenienze, avere vaste conseguenze i quali vengono sviluppati, utilizzati e migliorati nel corso degli anni.
Questi verrenno presi in considerazione solo per dare un'idea dell'importanza della programmazione, degli interessi che richiama e delle attività che muove.

B80w.04 Tutti i programmi si servono di manovre che sono richieste da molti altri programmi, ad esempio di manovre di lettura dei dati, di manovre di emissione dei risultati in forme leggibili e di esecuzioni di calcoli numerici, geometrici, statistici o simili che si diversificano solo in conseguenza dei diversi valori dei dati da manipolare.
Questi calcoli di interesse generale sono convenientemente affidati a sottoprogrammi che possono essere richiamati da grandi varietà di procedure più specifiche.
Attualmente sono disponibili grandi collezioni di sottoprogrammi chiamate librerie. In particolare il linguaggio C++ fa riferimento a molte di queste librerie.

Quando la stesura di un programma (di media complessità) richiede un testo di qualche migliaia di linee è opportuno che sia sviluppato in parti separate dedicate a sottocompiti parziali e con testi registrati su files simbolici diversi che chiamiamo **moduli di programma**.
I vantaggi di questa parcellizzazione sono molteplici, molti dei quali simili a quelli che si hanno con gran parte dei prodotti industriali.
Si possono mettere a punto le diverse parti separatamente, eventualmente attraverso il lavoro di più programmatori che lavorano con una certa indipendenza. Si possono avere librerie che affrontano problemi non solo general purpose, ma che riguardano interessi settoriali e che si evolvono anche nell'arco di molti anni in relazione all'evoluzione di un intero settore applicativo.

In tal modo si giunge alla cosiddetta programmazione modulare, che ha portato ad arricchire i linguaggi di programmazione e i loro sistemi di sviluppo.
Nel linguaggio si devono ave=re meccanismi che rendano agevole e sicuro il collegamento fra i vari moduli e i vari sottoprogrammi.
Per quanto riguarda la traduzione del testo sorgente per avvicinarsi alle istruzioni della macchina servono un programma compilatore, **compiler** per i sibgoli moduli e un programma collegatore, **linker**, in grado di collegare i diversi moduli compilati in un unico programma comprensibile al computer, il cosiddetto **programma eseguibile**.

Non entriamo nei dettagli di questi programmi, in quanto il programmatore li può ignorare: essi sonoo rganizzati in modo tale per cui il programmatore si può basare su uno scenario semplificato degli effetti delle richieste espresse in un programma che gli consente di controllare le manovre eseguite nei successivi passi delle possibili elaborazioni ottenibili in modo da poter governare il raggiungimento dei suoi obiettivi.

B80**w.05** La prima parte della traduzione di un modulo di programma nel linguaggio C++ consiste nel controllare che il testo soddisfi le regole formali del linguaggio e più dettagliatamente le regole del lessico e le regole della sintassi del linguaggio.
In queste fasi si possono trovare errori che rendono inaccettabile il modulo, errori che forse sono correggibili con interventi del traduttore e situazioni che fanno sospettare che il programmatore non si è espresso come avrebbe dovuto per ottenere un risultato ragionevole o che fanno suggerire modifiche migliorative.

In tutti i casi il compilatire fornisce segnalazioni diagnostiche che devono essere chiare, devono indicare i punti del testo che hanno portato alla diagnosi e possibilmente devono suggerire quali interventi correttivi possano migliorare il testo.

Nel caso di errore inaccettabile si ha l'arresto della traduzione.
Nel caso di errore che ha portato a una modifica giudicata ragionevole, nel caso di sospetto di errore e nel caso di suggerimenti miglirativi si hanno avvertimenti, warnings, che richiedono al programmatore di intervenire per una correzione oppure di autorizzare il proseguimento della traduzione del testo sospettato di imprecisione o di bassa efficienza.

B80**w.06** Quando tutti i moduli sono stati compilati il linker deve controllare che tutti i sottoprogrammi richiesti son disponibili nei moduli forniti dal programmatore o nelle librerie che sono state richieste con semplici linee dichiarative.
Se manca qualche sottoprogramma si ha l'arresto della elaborazione e si aspetta che questa lacuna venga colmata.
In caso contrario si procede alla alaborazione, eventualmente dopo una autorizzazione esplicita.

Le elaborazioni dei programmi, si parla anche di "corsa di un programma" e di "run time", possono svilupparsi nei modi più disparati.
Possono richiedere millisecondi o tempi lunghissimi. Possono richiedere interventi o concludersi senza richieste. Si possono avere segnalazioni di dettagli sulle manovre attuate o solo una segnalazione di elaborazione conclusa. Si possono avere segnalazioni da parte del programma di situazioni dubbie o inattese incontrate. Si possono avere seganalazioni di situazioni anomale o inaccettabili incontrate.

Inoltre può accadere che l'esecuzione prosegue senza emettere indicazioni sull'andamento della ricerca di risultati o di una soluzione lasciando il dubbio che l'esecuzione stia per concludersi in tempi superiori

ai previsti, oppure stia procedendo in elaborazioni inutili per qualche errore di previsione e quindi sia opportuno interromperla.
In questi casi la scelta di come uscire dal dilemma è a carico del programmatore o di un altro responsabile del programma.

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