

Programmi

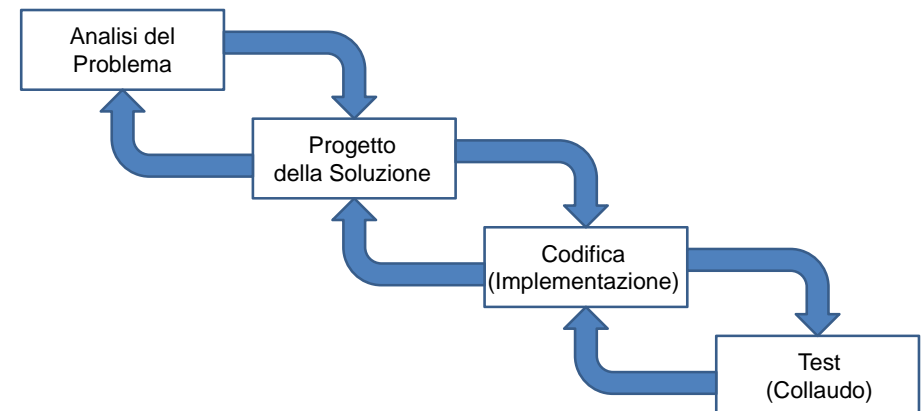
Sommario

- Obiettivo della programmazione e ciclo di sviluppo di programmi
- Istruzioni variabili e tipi
- Sottoprogrammi
- Strutture di controllo
- Ricorsione

Obiettivo

- Risoluzione di un problema mediante un calcolatore
 - Partire dalla descrizione del problema
 - in linguaggio naturale
 - per giungere alla stesura di un programma
 - nel linguaggio di programmazione scelto

Ciclo di sviluppo di programmi



Fasi di Sviluppo

- (Studio di fattibilità)
- Analisi
 - Chiarifica del problema
 - COSA si ha a disposizione ?
 - COSA si deve ottenere ?
- Progettazione
 - Individuazione di una strategia di soluzione
 - COME raggiungere l'obiettivo dato quello di cui si dispone ?
 - Scelta delle strutture di dati
- Codifica
 - Scrittura del programma
- Verifica (e correzione)
 - (Test) del programma
 - Rimanda ad una delle fasi precedenti
- Manutenzione
 - Correttiva, Adattativa, Migliorativa

Programmi

5

Sviluppo di un Programma (1)

- Accumulo degli errori
 - Gli errori in ciascuna fase si ripercuotono su tutte le fasi successive
 - Più costosi se rinvengono da fasi più lontane
 - Ciascuna fase può aggiungere errori a quelli delle fasi precedenti
- Vari approcci
 - Differente sequenza di esecuzione delle fasi di sviluppo

Programmi

6

Sviluppo di un Programma (2)

- Documentazione di ogni fase
 - Necessaria per chi riprenderà in seguito il programma per modificarlo
 - Gli autori stessi
 - Altri sviluppatori
- La documentazione prodotta in ciascuna fase rappresenta l'input per le fasi successive

Programmi

7

Comunicazione dell'algoritmo all'elaboratore

- Linguaggi non ambigui e sequenziali
 - Comprensibili alla macchina
- Requisiti per la descrizione
 - Univoca
 - Non dà adito ad interpretazioni errate
 - Completa
 - Prevede tutte le azioni necessarie
 - Ripetibile
 - Garantisce un buon risultato se eseguita da più esecutori con medesime caratteristiche

Programmi

8

Programmazione (1)

- Descrizione del procedimento di soluzione di un problema ad un esecutore meccanico
- Poiché l'esecutore è meccanico, consiste nel
 - Ricondurre il problema da risolvere a problemi primitivi
 - Eseguibili come insieme di azioni primitive
 - Organizzare ed utilizzare le "risorse" dell'elaboratore

Programmazione (2)

- Trasformazione della descrizione di un algoritmo in un messaggio
 - Insieme di istruzioni codificate in un linguaggio interpretabile da un esecutore
- Passa attraverso un'astrazione
 - Sia delle operazioni che il procedimento prevede
 - Sia degli oggetti su cui il procedimento deve operare

Programma (1)

- Traduzione della procedura di soluzione in un linguaggio comprensibile alla macchina con indicazioni sui dati di ingresso e uscita
- Comunica al calcolatore **istruzioni operative**
 - Quali dati di ingresso deve trattare
 - Come deve operare su questi dati
 - Quali dati deve dare come risultato

Programma (2)

- Procedura eseguibile su calcolatore, che rappresenta una soluzione ad un problema
 - Risultato di un lavoro di analisi e progetto che inizia dalla formulazione del problema
 - Corrisponde alla tripla (Dati, Algoritmo, Risultati)
 - [Wirth] Algoritmi + Strutture Dati = Programmi

Programma (3)

- Traduzione di un metodo di soluzione eseguibile in un linguaggio comprensibile alla macchina
 - Descrive come vanno elaborati insiemi di valori che rappresentano le entità del problema
 - Usa rappresentazioni simboliche per estendere l'applicabilità del metodo di soluzione a valori diversi
 - Uso di variabili

Dati

- Entità su cui lavora il programma
 - Costanti
 - Variabili
- Rappresentati come sequenze di bit
 - Nei linguaggi ad alto livello il programmatore può ignorare i dettagli della rappresentazione
 - Tipo di dato

Istruzioni

- Operative
 - Lavorano su rappresentazioni delle entità del problema
- Dichiarative
 - Consentono di definire come interpretare e rappresentare tali entità in termini di variabili nel programma
 - Totale caratterizzazione mediante la definizione di:
 - Un nome
 - Un tipo

Istruzioni Dichiarative

- Definiscono le aree di memoria in cui sono conservati i dati cui fa riferimento un algoritmo
 - Predispongono le posizioni di memoria da utilizzare
 - Associano un nome a ciascuna di esse
 - Identificatore
 - Determinano il tipo di dati che vi possono essere memorizzati
 - Insieme dei valori permessi
 - Insieme delle operazioni applicabili

Tipi di Istruzioni

- Un linguaggio di programmazione dispone di:
 - Istruzioni di ingresso
 - Permettono all'esecutore di conoscere informazioni fornite dall'esterno
 - Istruzioni di uscita
 - Permettono all'esecutore di notificare all'utente i risultati ottenuti dall'elaborazione
 - Istruzioni operative
 - Permettono di effettuare calcoli o, comunque, operazioni sulle entità astratte rappresentanti gli elementi del problema
 - Strutture di controllo

Istruzioni di Ingresso/Uscita

- Livello di descrizione dell'algoritmo
 - Necessità di indicare i dati su cui operare
- Livello di programma
 - Necessità di comunicare i dati e i risultati
 - Istruzioni di lettura e scrittura

Istruzioni di Uscita

- Consentono di notificare all'utente il valore di una variabile del programma
 - Visualizzazione, stampa, ...
- Attivano un'operazione di scrittura
 - Copiatura su un supporto esterno
 - carta, nastri e dischi magnetici, display, ...
 - del contenuto di un'area di memoria denotata dal nome della variabile che compare nell'istruzione di scrittura
- Esempio: print y

Istruzioni Dichiarative Variabili

- Forniscono una lista contenente i nomi scelti per le variabili e i tipi corrispondenti
 - Convenzione: indicare tutte le variabili
- Necessarie nei linguaggi di programmazione
 - Nel programma si fa riferimento agli indirizzi delle aree di memoria in cui sono conservati i dati

Variabile (1)

- Nome simbolico per denotare un'area di memoria e, tramite essa, il valore contenuto
 - Contiene una rappresentazione di un oggetto su cui l'algoritmo opera
 - Memorizzazione di un valore
 - Istruzioni di ingresso
 - Istruzioni di assegnamento

Variabile (2)

- Caratterizzata da
 - Un nome
 - Identificatore
 - Un valore
 - Un tipo
 - Attributo che specifica l'insieme di valori che la variabile può assumere

Variabile (3)

- Rappresenta una locazione di memoria del computer, contraddistinta da uno specifico indirizzo, che contiene il valore su cui applicare le istruzioni del programma
- Un identificatore denota una coppia
 - Posizione di memoria
 - Quantità in essa contenuta
 - Una limitazione nel rappresentare dati di tipo numerico o alfanumerico viene dalle dimensioni limitate della memoria

Tipo

- Attributo di una variabile che ne specifica ed individua:
 - L'insieme dei valori che la var. può assumere
 - L'insieme di operazioni effettuabili sulla var.
 - Il modo con cui ci si può riferire alla var.
- Per parlarne in termini formali è utile introdurre il concetto di algebra di dati
 - Esempio:
Stipendio: numero intero non negativo

Algebra di Dati

- Famiglia di insiemi
 - Insiemi di dati
- Famiglia di operatori sui dati
 - Operatori
- Repertorio di simboli per indicare l'insieme dei dati
 - Nomi
- Repertorio di simboli per indicare gli operatori
- Repertorio di simboli per indicare elementi singoli degli insiemi di dati

Programmi

25

Algebra di Dati: Esempio (1)

- Famiglia di insiemi
 - Numeri interi
 - Valori logici
- Famiglia di operatori
 - Operatori aritmetici
 - Somma, sottrazione, moltiplicazione, divisione, modulo
 - Operatori per i valori logici
 - Negazione, congiunzione, disgiunzione, ...
 - Operatori di confronto fra interi
 - Maggiore (stretto), minore (stretto), uguale, diverso

Programmi

26

Algebra di Dati: Esempio (2)

- Nomi
 - **int, boolean** per gli insiemi di dati
 - **+, -, *, /** per gli operatori aritmetici
 - **!, &&, ||** per gli operatori logici
 - **<, ==, >, <=, >=, !=** per i confronti fra interi
- Costanti
 - Ogni intero è indicato da una sequenza di cifre decimali, eventualmente precedute da + o -
 - Interpretata secondo la notazione decimale
 - I valori logici sono di solito indicati dalle costanti true e false

Programmi

27

Algebra di Dati

- Ogni linguaggio di programmazione propone
 - Dati
 - Operatori
 - Funzioni che
 - Prendono come argomenti dei dati
 - Restituiscono dati come risultati
 - Argomenti e risultati sono dati di tipo prestabilito all'atto della definizione dell'operatore

Programmi

28

Algebra di Dati

- Caratteristiche dipendenti da
 - la famiglia di insiemi
 - la famiglia di operatori
 - Possono essere combinati per ottenere espressioni
- Dunque:
 - Un dato indica un valore che una variabile può assumere
 - Un tipo di dato è un modello matematico che sta ad indicare una collezione di valori sui quali sono ammesse certe operazioni

Tipo

- Tripla
 $T = \langle D, C, O \rangle$
 - Dominio
 - Costanti
 - Operatori
 - Funzioni
 - Predicati

Tipo: Esempio

- Tipo dei complessi
 - Dominio
 - Sottoinsieme dei numeri complessi
 - Costanti
 - Parte reale
 - Parte immaginaria
 - Operazioni
 - Addizione $+$: (complesso X complesso) \rightarrow complesso
 - Non ci interessiamo della rappresentazione interna dei valori e delle operazioni

Dichiarazione di Tipo: Utilità

- Definizione del dominio di applicazione del programma
- Comprensione funzionamento dell'algoritmo
- Verifica correttezza del programma
 - Compilatore
 - Programmatore
- Definizione dello spazio di memoria necessario
- Rappresentazione interna
 - Esempio: 18 e 18.0

Definizione di Tipo Meccanismi Linguistici

- Istruzioni e costrutti linguistici necessari per informare l'esecutore su
 - dominio della variabile
 - insieme di operazioni effettuabili su di essa
 - modo attraverso cui ci si può riferire ad essa
- Esempio: Indicazione esplicita dei valori permessi per ogni variabile
 - Costanti di tipo

Definizione di Tipo nei Ling. di Programmazione

- I moderni linguaggi di programmazione mettono a disposizione
 - Un insieme di tipi di uso più comune
 - Tipi predefiniti
 - Gli strumenti per poter costruire qualunque tipo di dati

Tipi Standard

- Tipi più comuni di variabili
 - Interi
 - Reali
 - Logici
 - Caratteri
- Valori rappresentabili limitati
 - Dimensioni della memoria che dovrà ospitarne le variabili
 - Tipo numerico
 - Tipo alfanumerico

Tipo degli Interi (1)

- Valori nell'insieme dei numeri interi
 - Stringa di cifre, eventualmente preceduta dal segno
 - Positivi
 - Negativi
- Operazioni basilari
 - somma, prodotto, differenza, quoziente, resto, elevamento a potenza

Tipo degli Interi (2)

- Non sono in numero **infinito** nell'aritmetica dei calcolatori
 - Per ogni macchina esistono
 - il più grande intero
 - il più piccolo intero
 - rappresentabile in una locazione di memoria
- Alcune proprietà dell'aritmetica **non** restano valide in generale
 - Risultato di un'operazione non rappresentabile
 - Overflow

Tipo dei Reali (1)

- Valori nell'insieme dei numeri reali
 - Parte intera
 - Parte decimale
 - Differenza tra il numero e la sua parte intera (< 1)
 - Sequenza potenzialmente infinita di cifre
 - In alcuni casi esiste un'ultima cifra diversa da zero, seguita da una successione infinita di zeri

Tipo dei Reali (2)

- Non formano un **continuo** nell'aritmetica dei calcolatori
 - Ciascuno rappresenta un intervallo del continuo
 - Insieme di infiniti valori reali
 - Ad ogni numero reale è associata una rappresentazione
 - Intervallo in cui ricade
- Forma un sottoinsieme dei numeri reali

Tipo dei Reali (3)

- Rappresentazione ottenuta per troncamento o arrotondamento
 - Possibili errori di precisione anche consistenti
- Esiste un valore massimo
 - Overflow
 - Rappresentazione indefinita per tutti i valori maggiori

Tipo dei Valori Logici

- Rappresentano valori di verità
 - Falso, Vero
 - Falso < Vero
- Usati tipicamente nelle condizioni
 - Ottenibili come risultato di confronti

Tipo dei Valori Logici - Operatori

- Per priorità decrescente:

- not
- and
- or

<i>x</i>	<i>y</i>	<i>not x</i>	<i>x and y</i>	<i>x or y</i>
<i>F</i>	<i>F</i>	<i>V</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>V</i>	<i>V</i>	<i>F</i>	<i>V</i>
<i>V</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>V</i>
<i>V</i>	<i>V</i>	<i>F</i>	<i>V</i>	<i>V</i>

- Sottoinsieme completo
 - Può simulare tutti gli altri operatori logici

Tipo dei Caratteri

- Insieme finito ed ordinato di simboli
 - Lettere dell'alfabeto
 - Cifre decimali
 - Punteggiatura
 - Simboli speciali
 - Spaziatura (blank), Ritorno carrello, A capo, Separatore di linea (EOL), ...
- Costanti di tipo Carattere racchiuse tra apici
 - Esempi: 'a' '8' '?' '@'

Tipo dei Caratteri - Rappresentazione

- Corrispondenza biunivoca tra l'insieme dei caratteri e un sottoinsieme degli interi
 - Standard ASCII (American Standard Code for Information Interchange)
 - UNICODE

Tipo dei Caratteri - Proprietà

- Funzioni di trasferimento
 - ord(c)
 - numero d'ordine del simbolo c nella tavola di codifica
 - chr(i)
 - Simbolo il cui numero d'ordine è i
 - Proprietà
 - $\text{ord}(\text{chr}(i))=i$ $\text{chr}(\text{ord}(c))=c$
 - SE $c1 < c2$ ALLORA $\text{ord}(c1) < \text{ord}(c2)$
 - Relazione d'ordine totale
 - Coerente con i sottoinsiemi delle lettere e delle cifre

Tipo dei Caratteri - Note

- Alcuni caratteri non sono stampabili
 - Esempio: campanello (bell)
- NB: **caratteri delle cifre diversi dalle cifre**
 - Hanno come rappresentazione interna un valore diverso dalla cifra che rappresentano
- Relazione d'ordine totale
 - Proprietà riflessiva, antisimmetrica, transitiva

Istruzioni di Ingresso

- Permettono di acquisire informazioni dall'esterno, inserendole in opportune variabili del programma
- Attivano un'operazione di lettura
 - Assegnazione del valore letto su un supporto di memorizzazione esterno
 - schede, nastri e dischi magnetici, ...
 - ad un'area di memoria individuata dal nome che compare nell'istruzione di lettura
- Esempio: get x

Assegnazione di valori a variabili

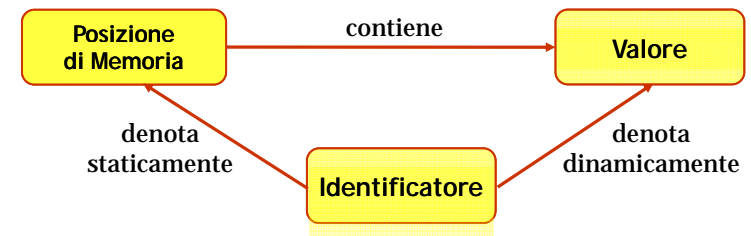
- Avviene in 2 passi
 - Produzione di un nuovo valore
 - Assegnamento di quel valore alla variabile
- L'operazione di assegnamento è indicata con simboli diversi a seconda dei diversi linguaggi di programmazione
 - := (Pascal)
 - = (C, C++, Java, ...)
 - ...
- Confusione fra uguaglianza e assegnazione

Assegnazione

- Per produrre il valore si possono usare espressioni (aritmetiche o logiche) il cui risultato è un singolo valore
- Il valore prodotto dall'espressione a destra viene memorizzato nell'area di memoria riservata alla variabile a sinistra
 - La memorizzazione del valore ottenuto nella locazione di memoria riservata alla variabile implicata nell'assegnamento rimpiazza qualunque valore contenuto in precedenza

Legami degli Identificatori

- Identificatore – Posizione di memoria
 - Statico
- Identificatore – Valore
 - Dinamico nel programma



Legami degli Identificatori

- La dinamicità del legame identificatore-valore consente di scrivere senza contraddizioni assegnazioni del tipo
- $x = x + 1$
 - Alla posizione di memoria identificata da x assegna il valore ottenuto calcolando la somma del valore già memorizzato e 1

Assegnazione Computo dei Valori

- Le operazioni di assegnazione possono implicare calcoli complessi
 - Espressioni aritmetiche
 - Espressioni logiche e predicati

Espressioni Aritmetiche

- Formate da associazioni di variabili e costanti secondo regole opportune e attraverso l'applicazione di definiti operatori numerici

Simbolo	Tipo di valore cui dà luogo	Operazione
+	Numerico	Somma
-, ...	"	Differenza
*, ^, ...	"	Prodotto
/, ÷, DIV,	"	Divisione
***, ^, ...	"	Potenza

Espressioni Logiche o Predicati

- Usate in:
 - Assegnazioni fatte ad una variabile logica
 - Strutture di controllo che comportano la verifica di condizioni

Simbolo	Tipo di valore cui dà luogo	Operazione
NOT, AND, OR	Logico	Connettivi logici
=	"	Uguaglianza
≠	"	Diversità
<	"	Minoranza
>	"	Maggioranza
≤	"	Minore o uguale
≥	"	Maggiore o uguale

Costanti

- Dati il cui valore viene definito inizialmente e non varia per tutta l'esecuzione del programma
 - Accessibili solo in lettura
- Garanzia nell'uso
 - Impossibile eseguire assegnazioni sugli identificatori corrispondenti

Strutture di Controllo

- Obbligatorie
 - Sequenza
 - Selezione binaria
 - Una iterazione illimitata
- Opzionali
 - Selezione multipla
 - L'altra iterazione illimitata
 - Iterazione limitata

Definizione di Tipo

- Fornisce, tramite le istruzioni dichiarative:
 - L'indicazione di tutti i valori che caratterizzano il tipo
 - L'eventuale strutturazione di insiemi di valori
- Esempio: tipo $t : T$
 - t è il nome che indica il tipo
 - T ne è la descrizione

Tipi Semplici

- I cui elementi sono singoli valori
 - primitivi (forniti dal linguaggio)
 - definiti dall'utente

Tipi Scalari

- Corrispondenza uno-a-uno fra i valori e un intervallo di interi
 - Ordinali dei relativi valori
- Operazioni consentite:
 - Confronto
 - Assegnamento
 - Funzioni predecessore-successore
 - $\text{pred}(X)$
 - $\text{succ}(Y)$

Tipi Scalari

- Sono tipi scalari quelli usati più di frequente
 - Solitamente predefiniti in ogni linguaggio
 - Interi, Logici, Caratteri
- Altri sono definibili dal programmatore
 - Enumerativi (il più spontaneo)
 - Elenco dei valori
 - Sottocampo di un tipo scalare
 - Valori estremi (min/max)

Dati Strutturati

- Insiemi di valori correlati
 - Indicati collettivamente da un unico nome
 - Si presuppone che tra essi esista una struttura legata
 - all'organizzazione
 - al tipo di valori che compongono l'insieme
 - alle operazioni per estrarre i dati dall'insieme
 - È fondamentale il modo in cui vengono individuati i dati componenti

Variabili Strutturate

- I cui valori sono aggregazioni di componenti
 - La dichiarazione prevede l'indicazione di
 - Nome
 - Tipo di struttura
- I linguaggi di programmazione offrono costruttori per le strutture più comuni
 - Operatori già definiti
 - Alcuni linguaggi consentono operazioni su intere variabili strutturate
- Strutture più complesse definibili attraverso il costrutto di tipo

Struttura di Dati

- Agglomerato (significativo) in cui sono riuniti dati da elaborare
 - Particolare tipo di dato
 - Caratterizzato più dall'organizzazione imposta agli elementi componenti che dal tipo degli elementi stessi
- Consiste di
 - Un modo sistematico di organizzare i dati
 - Un insieme di operatori per
 - Manipolare elementi della struttura
 - Aggregare elementi per costruire altre strutture

Strutture di dati Esempi

- Tabelle
 - Orario delle Lezioni
 - Orario Ferroviario
 - Matrici
- Date
- Schede
 - Patente e altri documenti di riconoscimento
 - Schede di biblioteca

Strutture di Dati

- I moderni linguaggi di programmazione mettono a disposizione
 - Un insieme di strutture di uso più comune (predefinite)
 - Sufficiente l'indicazione di
 - Dimensione
 - Tipo delle componenti
 - Gli strumenti per poter costruire qualunque tipo di struttura
 - Costruttori di tipo

Strutture di Dati Dimensioni di Classificazione

- Disposizione dei dati componenti
 - Lineari
 - Dati disposti in sequenza
 - Primo elemento, secondo elemento, ...
 - Non lineari
 - Non è individuata una sequenza
- Numero di dati componenti
 - A dimensione fissa
 - Il numero di elementi della struttura rimane costante nel tempo
 - A dimensione variabile
 - Il numero di elementi può variare nel tempo

Strutturare i Dati

- Applicazioni
 - Vettore/Matrice
- Prodotto Cartesiano
 - Record
- Insieme Potenza
 - Set
- Sequenze
 - File

Vettore

- Tabella monodimensionale
 - Struttura lineare
 - A dimensione fissa
- Sequenza di elementi dello stesso tipo
 - Operazioni consentite:
 - Lettura (selezione)
 - Reperimento del valore di un elemento
 - Scrittura (sostituzione)
 - Sostituzione del valore di un elemento con un nuovo valore

Vettore

- Numero fissato di componenti
 - Tutte dello stesso tipo
 - Tipo base
 - Ciascuna esplicitamente denotata ed indirizzata tramite un selettore (indice)
 - Non si è legati ad uno specifico tipo di indice
- Definito da:
 - Tipo degli elementi
 - Numero degli indici
 - Tipo degli indici

Programmi

69

Vettore

- vettore (tipo_indice) di tipo_base
- Accesso a qualunque componente
 - Specificandone la posizione
 - Nome della variabile vettore seguito dall'indice
 - In un tempo indipendente dal valore dell'indice
 - Accesso diretto (random)
- Un elemento di un vettore può essere a sua volta di un tipo strutturato

Programmi

70

Vettori Rappresentazione

- Componenti allocate in locazioni contigue
 - Ordinatamente
 - Consecutivamente
- Occupazione totale di memoria $d * n$
 - Tipo base d parole di memoria
 - Dipendente dal tipo di componenti
 - Vettore n elementi
- Posizione della prima componente: I_0
 - j -esima componente $I_0 + (j - 1) * d$

Programmi

71

Vettori

- Dipendono dal linguaggio di programmazione:
 - Modalità di
 - Dichiarazione
 - Definizione dell'intervallo di variabilità dell'indice
 - Scrittura degli indici
 - Possibilità di operazioni multiple
 - Agiscono su tutti gli elementi della struttura
 - Purché abbiano uguali dimensioni
 - Se non definite, necessarie iterazioni

Programmi

72

Vettori Esempio

- In matematica, matrici
 - A una dimensione (vettori) o a più dimensioni
 - Sono fondamentali le dimensioni
 - Variabili sottoscritte o con indici
 - Considerate come un tutto unico
 - Operazioni tra matrici (algebra matriciale)
 - Indici interi
 - Rappresentano la posizione che quella variabile occupa in una struttura di variabili
 - Servono ad indicare univocamente quella variabile

Record - Struct

- Registra in una n-pla di dati le principali caratteristiche di un'entità
 - Struttura non lineare
 - A dimensione fissa
- Insieme di dati non omogenei
 - Operazioni consentite:
 - Lettura (selezione)
 - Reperimento del valore di un elemento
 - Scrittura (sostituzione)
 - Sostituzione del valore di un elemento con un nuovo valore

Record - Struct

- Numero fissato di componenti
 - Tipi potenzialmente diversi
 - Ciascuna esplicitamente denotata ed indirizzata tramite un selettore (campo)
 - Paragonabile ad una variabile ordinaria
 - Denotato da un identificatore
- Definito dalla descrizione, per ogni singola componente, di:
 - Tipo
 - Limiti di variabilità del valore che può assumere
 - Identificatore per accedervi

Record - Struct

- record identificatore : tipo;
- ...
- identificatore : tipo
- Accesso a qualunque componente
 - Specificandone il campo
 - Nome della variabile record seguito dall'identificatore del campo
- Un elemento di un record può essere a sua volta di un tipo strutturato

Record - Struct Rappresentazione

- Componenti allocate
 - In posizioni di memoria contigue
 - Nell'ordine in cui sono specificate nella dichiarazione
- Occupazione di memoria complessiva
 - Somma dell'occupazione di ciascun campo
 - Note in fase di compilazione
- Posizione di memoria di un campo
 - Somma della posizione iniziale del record e della somma delle dimensioni dei campi precedenti

81

Record - Struct

- Variabile strutturata a molte componenti
 - Aggregazioni di dati
 - Tipi potenzialmente differenti
 - Accesso alle componenti tramite nome
- Astrazione delle modalità di memorizzazione dei dati usate a livello di linguaggio macchina

Programmi

82

Record - Struct Esempi

- Data
 - Giorno, Mese, Anno
- Scheda bibliografica
 - Autore, Titolo, Prezzo, Anno, Prestito
- Indirizzo
 - Via, N. civico, CAP, Città, Provincia
- Scheda anagrafica
 - Nome, Cognome, Data di nascita, Stato civile

Programmi

83

Array vs. Record

- | | |
|--|---|
| <ul style="list-style-type: none">• Dimensione fissa• Tipo componenti omogeneo• Sequenza<ul style="list-style-type: none">– Ordine• Accesso diretto<ul style="list-style-type: none">– Indice<ul style="list-style-type: none">• Uso di espressioni<ul style="list-style-type: none">– Flessibilità | <ul style="list-style-type: none">• Dimensione fissa• Tipi componenti diversi<ul style="list-style-type: none">– Più generale• Insieme• Accesso diretto<ul style="list-style-type: none">– Identificatore di campo |
|--|---|

Programmi

84

Array & Record

- Spesso si ha a che fare con strutture formate da array di record
 - Sequenza di schede
 - Simili fra loro
 - Distinguibili in base alla chiave: un campo (o sottoinsieme dei campi) che identifica univocamente un record
 - Esempi
 - Schedario di dipendenti (Chiave: Codice Fiscale)
 - Orario Ferroviario (Chiave: Numero Treno)

Programmi

85

Programmazione Modulare (1)

- Tecnica basata sul metodo di scomposizione di un problema in sottoproblemi logicamente indipendenti tra loro
 - Ad ogni sottoproblema corrisponde un modulo
 - Codificati separatamente
 - Compilati separatamente (talvolta)
 - Integrati solo alla fine per formare il programma complessivo

Programmi

86

Programmazione Modulare (2)

- Un problema caratterizzato da
 - un algoritmo A
 - che opera sull'insieme dei dati di partenza D
 - per produrre l'insieme dei risultati R
- viene suddiviso in un insieme finito di n problemi a differenti livelli caratterizzati dalla tripla: (D_i, A_i, R_i)
 - Interazione e ordine di esecuzione degli algoritmi secondari per ottenere la soluzione del problema originario gestita da un algoritmo coordinatore

Programmi

87

Programmazione Modulare (3)

- Algoritmo coordinatore = programma principale
- Algoritmi secondari = sottoprogrammi
 - Diversi livelli
 - Gerarchia di macchine astratte, ciascuna delle quali
 - Realizza un particolare compito in modo completamente autonomo
 - Proprie definizioni di tipi, dichiarazioni di variabili e istruzioni
 - Fornisce la base per il livello superiore
 - Si appoggia su un livello di macchina inferiore (se esiste)

Programmi

88

Programmazione Modulare Tecniche

- Basate sul metodo di soluzione di problemi consistente nello scomporre un problema in sottoproblemi più semplici
 - Sviluppo top-down
 - Approccio step-wise refinement
 - Sviluppo bottom-up
 - Sviluppo “a sandwich”

Raffinamenti Successivi

- Metodo Basato su
 - Raffinamento di un passo della procedura di soluzione
 - Legato alle modalità di esecuzione conseguenti una certa suddivisione in sottoproblemi
 - Necessario concentrarsi sul “cosa” piuttosto che sul “come”
 - Raffinamento della descrizione dei dati
 - Definizione della struttura e tipo
 - Definizione delle modalità di comunicazione
 - Come renderli comuni a più sottoproblemi

Sviluppo Top-Down

- Costruzione del programma per livelli successivi
 - Corrispondenza con la scomposizione del problema cui è relativo
 - Strumento concettuale per la costruzione di algoritmi
 - Dettaglio successivo delle parti in cui viene scomposto (sottoprogrammi) fino al codice finale
 - Strumento operativo per l'organizzazione e lo sviluppo di programmi complessi
- Metodo trial and error
 - Prova e riprova alla ricerca della scomposizione ottimale

Sviluppo Bottom-Up

- Partendo dalle istruzioni del linguaggio
 - Costruzione di programmi molto semplici
 - Collegamento successivo in programmi più complessi
- fino ad ottenere il programma finale
- Usato soprattutto nell'adattamento di algoritmi codificati già esistenti a nuove situazioni

Metodo a Sandwich

- Basato su una cooperazione fra le tecniche top-down e bottom-up
 - Necessità di raffinare via via la soluzione del problema principale
 - Scomposizione in algoritmi che ne risolvono delle sottoparti
 - Disponibilità di algoritmi di base per problemi semplici
 - Raggruppamento in algoritmi via via più complessi

Sottoprogramma

- Insieme di istruzioni
 - Individuate da un nome
 - Che concorrono a risolvere un problema
 - Ben definito
 - Sensato
 - Non necessariamente fine a sé stesso
 - Supporto per la risoluzione di problemi più complessi
- Esempi:
 - Scambio, Ricerca del Minimo, Ordinamento, ...

Sottoprogrammi

- Un programma viene strutturato in sottoprogrammi:
 - Per rispettare la decomposizione ottenuta con il metodo di progettazione dell'algoritmo
 - Per strutturarne in maniera chiara l'architettura
 - Perché lo stesso tipo di elaborazione deve essere ripetuto più volte in diversi punti del programma

Sottoprogrammi Utilità

- Necessità di risolvere uno stesso problema
 - Più volte
 - All'interno dello stesso programma
 - In programmi diversi
 - Su dati eventualmente diversi
- Unicità dello sforzo creativo
 - Modifiche
 - Riutilizzo

Astrazioni Funzionali

- Fornite dai linguaggi di programmazione ad alto livello
 - Consentono di creare unità di programma
 - Dando un nome ad un gruppo di istruzioni
 - Stabilendo le modalità di comunicazione tra l'unità di programma creata ed il resto del programma in cui essa si inserisce
 - Assumono nomi diversi a seconda del linguaggio di programmazione
 - Subroutine
 - Procedure
 - Metodi
 - ...

Programmi

97

Astrazioni Funzionali

- Paragonabili a nuove istruzioni che si aggiungono al linguaggio
 - Definite dall'utente
 - Specifiche per determinate applicazioni o esigenze
 - Più complesse delle istruzioni base del linguaggio
 - Analogia con il rapporto fra linguaggi ad alto livello e linguaggio macchina
 - Ciascuna risolve un ben preciso problema o compito
 - Analogia con un programma

Programmi

98

Astrazioni Funzionali

- Struttura risultante di un programma:
 - Intestazione di programma
 - Definizione di tipi
 - Dichiarazioni di variabili
 - Dichiarazione di macchine astratte
 - Corpo di istruzioni operative
- Le dichiarazioni di macchine astratte rispecchiano le regole di struttura di un programma

Programmi

99

Sottoprogramma

- Indipendentemente dalle regole sintattiche del particolare linguaggio di programmazione
 - Individuabile con un nome
 - Identificatore
 - Prevede l'uso di un certo insieme di risorse
 - Variabili, costanti, ...
 - Costituito da istruzioni
 - Semplici o, a loro volta, composte
 - Differisce da un programma nelle istruzioni di inizio
 - Specificano che (e come) altri pezzi di programma possono sfruttarlo

Programmi

100

Chiamata di Sottoprogrammi

- Esecuzione delle istruzioni di un sottoprogramma
 - Modalità: deve essere comandata dal programma chiamante
 - Specifica del nome associato
 - Effetto: si comporta come se il sottoprogramma fosse copiato nel punto in cui è stato chiamato
 - Eliminazione di ridondanza

Chiamata di Sottoprogrammi

- All'atto dell'attivazione (su chiamata) dell'unità di programma
 - Viene sospesa l'esecuzione del programma (o unità) chiamante
 - Il controllo passa all'unità attivata
- All'atto del completamento della sua esecuzione
 - L'attivazione termina
 - Il controllo torna al programma chiamante

Sottoprogrammi

- Definizione
 - Titolo o intestazione
 - Identificatore
 - Specifica delle risorse usate
 - Corpo
 - Sequenza di istruzioni denotata dal nome della procedura
- Comunicazione
 - Come si connettono i sottoprogrammi tra di loro?
 - Come si scambiano dati?
 - Come comunicano col programma principale?

Sottoprogrammi Nidificazione

- Le risorse di cui fa uso un sottoprogramma possono includere altri sottoprogrammi
 - Completa analogia con i programmi
- Si viene a creare una gerarchia di sottoprogrammi
 - Struttura risultante ad albero
 - Relazione padre-figlio riferita alla dichiarazione

Sottoprogrammi Comunicazione

- Un sottoprogramma può comunicare
 - Con l'ambiente esterno
 - Attraverso istruzioni di lettura e/o scrittura
 - Con l'ambiente chiamante
 - Implicitamente
 - Operando direttamente sulle stesse variabili
 - Esplicitamente
 - Attraverso l'uso di parametri
 - » Evidenziano le variabili che il sottoprogramma ha in input e, opportunamente elaborate, vengono tramutate in output del sottoprogramma

Vista

- Insieme di risorse cui il sottoprogramma ha accesso
 - Dati
 - Altri sottoprogrammi
- Definita da
 - Nidificazione nella dichiarazione dei sottoprogrammi
 - Statica
 - Sequenza di chiamata dei sottoprogrammi
 - Dinamica
- Utile per limitare l'accesso alle risorse ai soli interessati

Vista Sottoprogrammi

- Ciascun sottoprogramma può richiamare solo i sottoprogrammi
 - Che esso dichiara direttamente
 - Dichiarati dallo stesso sottoprogramma che lo dichiara
 - Incluso se stesso
 - Ricorsione
- Visibilità definita esclusivamente in base alla nidificazione

Vista Variabili

- Ciascun sottoprogramma può usare esclusivamente
 - Le proprie variabili
 - Le variabili dichiarate dai sottoprogrammi attualmente in esecuzione
 - Visibilità dipendente
 - Dalla struttura della gerarchia di dichiarazione dei sottoprogrammi (statica)
 - Dall'ordine di chiamata dei sottoprogrammi precedenti (dinamico)

Vista: Shadowing

- Può capitare che diversi sottoprogrammi dichiarino risorse con lo stesso nome
 - Oggetti diversi, totalmente scorrelati
 - Possono avere tipi diversi
- Può capitare che, fra i sottoprogrammi attivi in un certo istante, alcuni abbiano dichiarato risorse con lo stesso nome
 - Ciascun sottoprogramma attivo ha accesso solo al sinonimo “più vicino”
 - Visibilità dipendente esclusivamente dall’ordine di chiamata dei sottoprogrammi precedenti

Programmi

109

Sottoprogrammi Tipi di Variabili

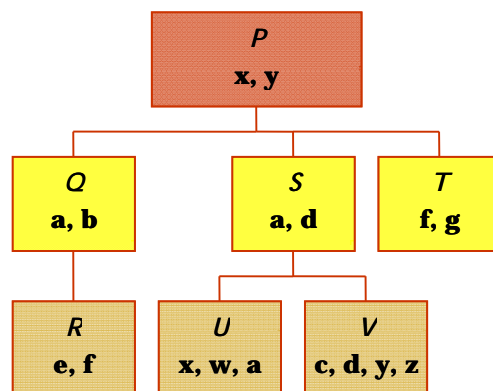
- Variabili locali
 - Interne al sottoprogramma
 - Temporanee
 - Create quando il sottoprogramma entra in azione
 - Distrutte quando il sottoprogramma è stata eseguita
 - Liberazione del relativo spazio di memoria
- Variabili non locali
 - Definite nel resto del programma, al di fuori del sottoprogramma
 - Dette globali se definite nel programma principale

Programmi

110

Vista Esempio

Unità di Programma P



Programmi

111

Effetti Collaterali

- Effetti di un sottoprogramma che altera il valore di una variabile non locale
 - La presenza di tali variabili impedisce che il sottoprogramma possa essere considerato come un’entità completa e autoconsistente
 - Non si riferisce esclusivamente alle sue costanti, variabili e parametri
 - Valutare attentamente l’uso, all’interno di un sottoprogramma procedura, di variabili non locali
 - Chiarezza
 - Sicurezza

Programmi

112

Parametri Formali

- Segnaposto utili ad indicare simbolicamente
 - gli oggetti su cui il sottoprogramma lavora
 - il loro tipo e struttura
- I loro nomi appaiono nell'intestazione della procedura
 - Non hanno alcuna connessione con nomi usati altrove
- All'atto della chiamata vengono sostituiti dai parametri effettivi (o reali)
 - Dati su cui effettivamente il sottoprogramma deve operare

Programmi

113

Parametri Formali

- Specificati all'atto della definizione del sottoprogramma
 - Legati al sottoprogramma
 - Simbolici
 - Consentono di definire
 - Quale tipo di dato deve essere passato alla procedura
 - Quale argomento deve essere trasmesso alla funzione
 - quando queste sono invocate

Programmi

114

Parametri Effettivi (actual parameters)

- Alla chiamata di un sottoprogramma, si specificano i dati effettivi su cui esso dovrà operare
 - valori costanti, variabili, espressioni, ...
 - conformità con i parametri formali
 - Numero, tipo e ordine
- L'esecuzione dell'istruzione di chiamata (o invocazione) comporta la sostituzione dei parametri formali con quelli reali

Programmi

115

Parametri Tipi di Passaggio

- La sostituzione può essere:
 - Per valore
 - Si calcola il valore del parametro reale e lo si sostituisce al corrispondente parametro formale (assegnazione)
 - Per riferimento
 - Il parametro effettivo è una variabile ed ha a disposizione una locazione di memoria il cui indirizzo viene "passato" al parametro formale
 - Per nome (o valore-risultato)
 - Il nome del parametro formale, all'occorrenza, viene sostituito col nome del parametro reale

Programmi

116

Passaggio di Parametri per Valore

- Generalmente usato per parametri che
 - Rappresentano un argomento di una funzione/sottoprogramma e non il risultato
 - Inutile (anzi potenzialmente dannoso) consentirne la modifica

Passaggio di Parametri per Riferimento

- Usato più spesso quando il parametro
 - rappresenta un risultato
 - necessità di conoscere la modifica
 - ha dimensioni notevoli
 - computazionalmente costoso ricopiarlo interamente
- Minore leggibilità
 - stessa variabile usata sotto diverse denominazioni
- Potenziale fonte di errori
 - errori nel sottoprogramma non più recuperabili

Funzioni

- Sottoprogrammi che hanno come risultato il calcolo di un valore
 - hanno un'istruzione che ritorna il risultato
- L'intestazione di funzione prevede una lista di parametri formali
- String cercaNome(Studente[] v, int matricola)
 - Nome della funzione: cercaNome
 - Il designatore di funzione specifica l'attivazione della funzione
 - Lista di parametri: (Studente[] v, int matricola)
 - Tipo del risultato: String

Funzioni dichiarazione

- Prevede un costrutto linguistico del tipo
- funzione <id> <lista-argomenti> : <tipo-risultato>
 - In <lista-argomenti> si rappresentano i parametri di entrata (con i loro tipi)
 - Come tali, teoricamente andrebbero sempre passati per valore al fine di evitare effetti collaterali
 - Come per le procedure, è possibile avere nella funzione una sezione delle dichiarazioni delle variabili locali

Sottoprogrammi come Parametri

- Nella classe dei parametri di sottoprogrammi rientrano anche altri sottoprogrammi
 - Usare un sottoprogramma F come parametro di sottoprogramma G:
F eseguito durante l'esecuzione di G
- Il parametro formale corrispondente ad un sottoprogramma
 - Riporta l'intestazione
 - I nomi del sottoprogramma e dei parametri possono cambiare
 - Vincolo: deve avere esclusivamente parametri passati per valore

Programmi

121

Sottoprogrammi come Parametri

- All'invocazione di un sottoprogramma avente come parametri altri sottoprogrammi
 - parametro effettivo = identificatore (o riferimento) relativo a un sottoprogramma avente i medesimi requisiti riguardo a parametri o tipo del risultato
 - durante l'esecuzione del sottoprogramma invocato, ogni occorrenza del parametro formale implica l'uso corrispondente del sottoprogramma fornito come parametro effettivo

Programmi

122

Ricorsione

- Si dimostra che
 - Ogni problema ricorsivo è computabile per mezzo di un programma
- e, viceversa,
 - Ogni problema computabile per mezzo di un programma è esprimibile in forma ricorsiva
- Le scomposizioni ricorsive implicano, a livello di codice, programmi in grado di invocare se stessi
 - procedure o funzioni ricorsive

Programmi

123

Ricorsione Esempio

- Calcolo del fattoriale
- $n! = n * (n - 1) * (n - 2) * \dots * 2 * 1$
 - esprimibile ricorsivamente come

$$\begin{array}{l} - \\ - \\ - \end{array} n! = \begin{cases} 1 & \text{se } n = 0 \\ n * (n - 1)! & \text{se } n > 0 \end{cases}$$

Programmi

124

Ricorsione Esempio

- Sia $nfatt(n)$ la funzione che calcola il fattoriale di n per ogni intero $n \geq 0$
 - Se $n = 0$
 - allora $nfatt(n) = 1$
 - altrimenti $nfatt(n) = n * nfatt(n - 1)$
- L'esecuzione della funzione $nfatt$ causa chiamate ricorsive della stessa funzione
 - Sovrapposizione di ambienti di programmazione nello stack in fase di esecuzione

Programmi

125

Ricorsione Esempio

- $4!$: poiché $4 > 0$, $4! = 4 * 3! = 4 * ?...$
 - $3!$: poiché $3 > 0$, $3! = 3 * 2! = 3 * ?...$
 - $2!$: poiché $2 > 0$, $2! = 2 * 1! = 2 * ?...$
 - $1!$: poiché $1 > 0$, $1! = 1 * 0! = 1 * ?...$
 - » $0!$: è noto che $0! = 1$
 - ... = $1 * 1 = 1$
 - ... = $2 * 1 = 2$
 - ... = $3 * 2 = 6$
- ... = $4 * 6 = 24$

Programmi

126

Mutua Ricorsione

- Insieme di sottoprogrammi che si chiamano a vicenda, l'ultimo dei quali chiama il primo
 - A chiama B che chiama C ... che chiama Z che chiama A
- Caso più semplice
 - 2 sottoprogrammi
 - A chiama B
 - B chiama A

Programmi

127

Mutua Ricorsione

- È un caso (più complesso) di ricorsione
 - La chiamata di un sottoprogramma a se stesso avviene attraverso una serie di chiamate intermedie
- Correttezza dipendente da 2 fattori
 - È necessario un livello assiomatico
 - Soluzione nota senza chiamata ricorsiva
 - Bisogna assicurarsi che la chiamata ricorsiva avvenga su un problema di ordine inferiore
 - Più vicino alla base

Programmi

128

Mutua Ricorsione Esempio

- Definizione dei numeri pari e dispari
 - Base: 0 è un numero pari
 - Passo:
 - Il successore di un numero pari è dispari
 - Il successore di un numero dispari è pari
- Ricorsiva
 - pari definito in termini di dispari, definito a sua volta in termini di pari
 - Parità di un numero (pari o dispari) definita in termini di proprietà del suo predecessore
 - Avvicinamento progressivo al caso noto (0)

Programmi

129



Torri di Hanoi

- Dati
 - 3 pioli
 - n dischi, tutti di diametro diverso, impilati su un piolo per diametro decrescente
- Obiettivo: Spostare la pila di dischi da un piolo a un altro
- Vincoli
 - Si può spostare un solo disco per volta
 - Qualunque impilamento di dischi su un piolo deve rispettare l'ordinamento per diametro decrescente

Programmi

130

Torri di Hanoi Soluzione Iterativa

- Finché non raggiungi la configurazione finale
 - Determina il disco da muovere
 - Determina il piolo su cui metterlo
 - Effettua lo spostamento
- Lo spostamento è un'azione primitiva
 - Per la scelta del piolo, notiamo che
 - I dischi di ordine pari percorrono i pioli in ordine alfabetico
 - Quelli di ordine dispari in senso contrario

Programmi

131

Torri di Hanoi Soluzione Iterativa

- Determina il disco da muovere
 - Se i dischi sono tutti in un piolo si muove quello in alto
 - Se i dischi sono tutti in 2 pioli non si muove il disco mosso per ultimo
 - Se i dischi sono in tre pioli si muove il più piccolo disco che non è stato mosso per ultimo
- Determina il piolo su cui metterlo
 - Determina se il disco è di ordine pari o dispari
 - Se è pari spostare il disco nel piolo che segue in ordine alfabetico
 - Se è dispari spostarlo sul piolo che segue in un ordine antialfabetico

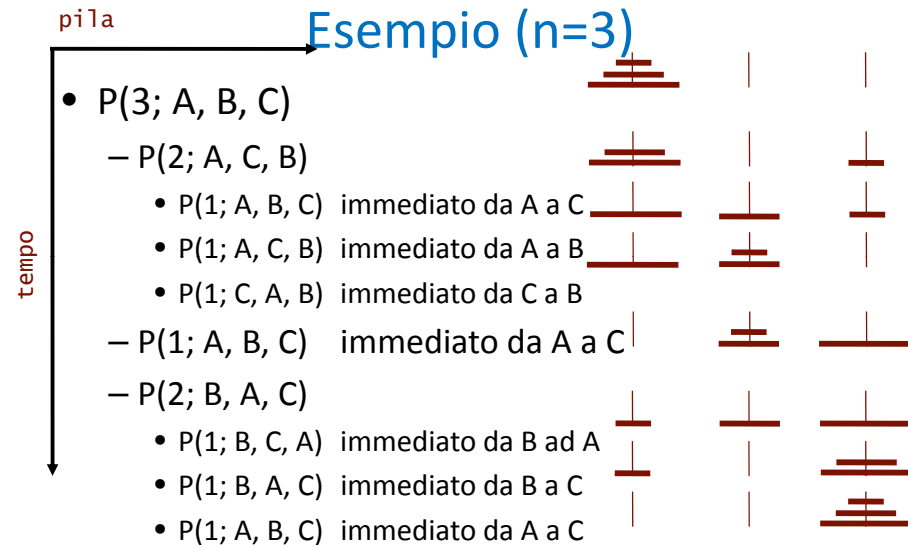
Programmi

132

Torri di Hanoi Soluzione Ricorsiva

- Notazione $P(n, \text{partenza}, \text{servizio}, \text{arrivo})$
 - Sposta n dischi dal piolo partenza al piolo arrivo usando il piolo servizio come appoggio
 - Problema iniziale: $P(n; A, B, C)$
- Se $n = 1$: muovi il disco da partenza ad arrivo
- Se $n > 1$:
 - Sposta $n - 1$ dischi da partenza a servizio usando arrivo come appoggio
 - Sposta il disco rimanente da partenza ad arrivo
 - Sposta gli $n - 1$ dischi da servizio ad arrivo usando partenza come appoggio

Torri di Hanoi Esempio ($n=3$)



Allocazione Dinamica della Memoria

- Consiste nell'allocare memoria durante l'esecuzione invece che all'atto della traduzione
 - Il compilatore alloca una quantità fissata di memoria per contenere l'indirizzo della componente allocata dinamicamente invece che la componente stessa
 - Si supera il problema della contiguità delle locazioni relativa ad una medesima struttura o programma
 - La memoria, così, è allocata automaticamente non appena la componente è referenziata

Allocazione Dinamica della Memoria

- Tecnica basata sull'uso di un puntatore
 - Variabile il cui valore è il nome di un'altra variabile creata dinamicamente
- Operazioni disponibili sul puntatore p
 - Accesso alla locazione il cui indirizzo è in p
 - Richiesta di nuova locazione e memorizzazione di tale indirizzo in p
 - Rilascio della locazione il cui indirizzo è in p

Tipo Puntatore

- Variabile contenente un indirizzo di memoria
 - Indirizzamento indiretto
- Necessità di specificare il tipo di dato che sarà contenuto a quell'indirizzo
 - Riservare lo spazio di memoria

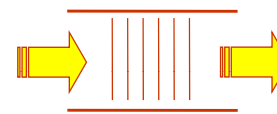
Sequenze in Memoria Centrale

- Pila (Stack): Sono possibili solo operazioni di accodamento e di estrazione dell'ultimo elemento inserito. Last In, First Out (LIFO)
- Coda (Queue): Comportamento tale che il primo elemento inserito è il primo ad essere elaborato. First In, First Out (FIFO)
- Lista concatenata (Linked List): Ciascun elemento conosce la posizione del successivo
 - Generico elemento raggiungibile tramite scansione



Pila Operatori

- Push(x,e)
 - Impilamento di un elemento e alla pila x
- Pop(x)
 - Estrazione dell'elemento in cima alla pila x
- Top(x)
 - Lettura dell'elemento in cima alla pila x
 - Non lo rimuove
- Empty(x)
 - Funzione booleana che controlla se la pila x è vuota



Coda Operatori

- Enqueue(x,e)
 - Accodamento di un elemento e alla coda x
- Dequeue(x,b)
 - Estrazione dell'ultimo elemento della coda x
- Empty(x)
 - Funzione booleana che controlla se la coda x è vuota

Liste Concatenate



- Memorizzata la posizione del primo elemento
- Ciascun elemento composto da due parti
 - Dato vero e proprio
 - Posizione dell'elemento successivo
- Ultimo elemento segnalato da un terminatore