

Corso di Programmazione

Sottoprogrammi

Procedure e funzioni

Prof.ssa Teresa Roselli
`roselli@di.uniba.it`

Programmazione Modulare

- 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

Programmazione Modulare

- 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 sottoproblemi 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

Programmazione Modulare

- Algoritmo coordinatore → programma principale o main
- Algoritmi secondari → sottoprogrammi
 - Diversi livelli
 - Si costruisce una 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)

Il programma è visto come un nuovo operatore disponibile sui dati

L'astrazione funzionale è la tecnica che permette di ampliare il repertorio di operatori disponibili

Programmazione Modulare

Tecniche per individuare i sottoproblemi

- 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”

Raffinamento per passi successivi

- 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 complessifino 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

- Corrisponde all'algoritmo secondario che risolve un sottoproblema
- Insieme di istruzioni
 - Individuate da un nome
 - Che concorrono a risolvere un problema
 - Ben definito
 - Sensato
 - Non necessariamente fine a se stesso
 - è di supporto per la risoluzione di problemi più complessi
 - rappresenta una funzionalità a se stante, una unità concettuale con un significato più ampio (prescinde dal problema presente)
- Esempi:
 - Scambio, Ricerca del Minimo, Ordinamento, ...

Sottoprogrammi

Utilità

- Un programma viene strutturato in sottoprogrammi:
 - Per rispettare la decomposizione ottenuta con il metodo di progettazione dell'algoritmo
 - Per strutturare in maniera chiara l'architettura del programma
 - Perché lo stesso gruppo di istruzioni deve essere ripetuto più volte in diversi punti del programma (blocchi ripetibili)

Sottoprogrammi

Utilità

- Risponde alla necessità di risolvere uno stesso problema
 - Più volte
 - All'interno dello stesso programma
 - In programmi diversi
 - Su dati eventualmente diversi
- Unicità dello sforzo creativo

Sottoprogrammi

Utilità

- Stile e qualità del software
 - Leggibilità
 - Manutenibilità
 - Trasportabilità
 - Modularità
 - Reuso

I sottoprogrammi giocano un ruolo fondamentale nella tecnica della programmazione

Sottoprogrammi

- SOTTOPROGRAMMA è una astrazione funzionale che consente di individuare gruppi di istruzioni che possono essere invoke esplicitamente e la cui chiamata garantisce che il flusso di controllo ritorni al punto successivo all'invocazione

Astrazioni Funzionali

- Fornite dai linguaggi di programmazione ad alto livello
 - Consentono di creare unità di programma (macchine astratte)
 - 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
 - Sub program
 - ...

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

Astrazioni Funzionali

- Struttura risultante di un programma:

Intestazione di programma

Definizione di tipi

Dichiarazioni di variabili

Dichiarazione di macchine astratte (sottoprogrammi)

Corpo di istruzioni operative del programma
principale

- La dichiarazione di una macchina astratta rispecchia le regole di struttura di un programma

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 (altre macchine astratte)
 - Differisce da un programma nelle istruzioni di inizio
 - Specificano che (e come) altri pezzi di programma possono utilizzarlo

Chiamata di Sottoprogrammi

- Provoca l'esecuzione delle istruzioni del 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

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

- Definizione
 - Titolo o intestazione
 - Identificatore
 - Specificazione delle risorse usate (talvolta)
 - Corpo
 - Sequenza di istruzioni denotata dal nome del sottoprogramma
- Comunicazione
 - Come si connettono i sottoprogrammi tra di loro?
 - Come si scambiano dati?
 - Come comunicano col programma principale?

Sottoprogrammi

Comunicazione

- Un sottoprogramma può comunicare
 - Con l'ambiente esterno
 - Istruzioni di lettura e/o scrittura
 - Con l'ambiente chiamante
 - Implicitamente
 - Tramite le variabili non locali (secondo le regole di visibilità del linguaggio)
 - Esplicitamente
 - Attraverso l'uso di parametri
 - » rappresentano le variabili che il sottoprogramma ha in input dal programma chiamante e che, opportunamente elaborate, vengono tramutate in output del sottoprogramma

Vista di un Sottoprogramma

- Rappresenta l'insieme delle risorse a cui il sottoprogramma ha accesso
 - Dati
 - Altri sottoprogrammi
- E' definita da
 - Nidificazione nella dichiarazione dei sottoprogrammi
 - *Vista Statica*
 - Sequenza di chiamata dei sottoprogrammi
 - *Vista Dinamica*
- Utile per limitare l'accesso alle risorse soltanto ai sottoprogrammi interessati

Vista di un Sottoprogramma

Sottoprogrammi

- Un sottoprogramma può richiamare soltanto i sottoprogrammi
 - che esso dichiara direttamente
 - che sono stati dichiarati dallo stesso sottoprogramma che lo dichiara
 - Incluso se stesso
 - Ricorsione
- Visibilità definita esclusivamente in base alla nidificazione
 - Figli e fratelli nella struttura ad albero

Vista di un Sottoprogramma

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 di un Sottoprogramma

Shadowing (oscuramento)

- Sottoprogrammi diversi possono dichiarare risorse con lo stesso nome
 - Oggetti diversi, totalmente scorrelati
 - Possono essere di tipi differenti
- Sottoprogrammi attivi in un certo istante possono aver 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

Sottoprogrammi

Tipi di Variabili

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

Sottoprogrammi

Tipi di Variabili

MAIN

Risorse globali

SOTTOPROGRAMMA P1

Risorse locali a P1 e non locali a P1.1

SOTTOPROGRAMMA P1.1

Risorse locali a P1.1

Sottoprogrammi

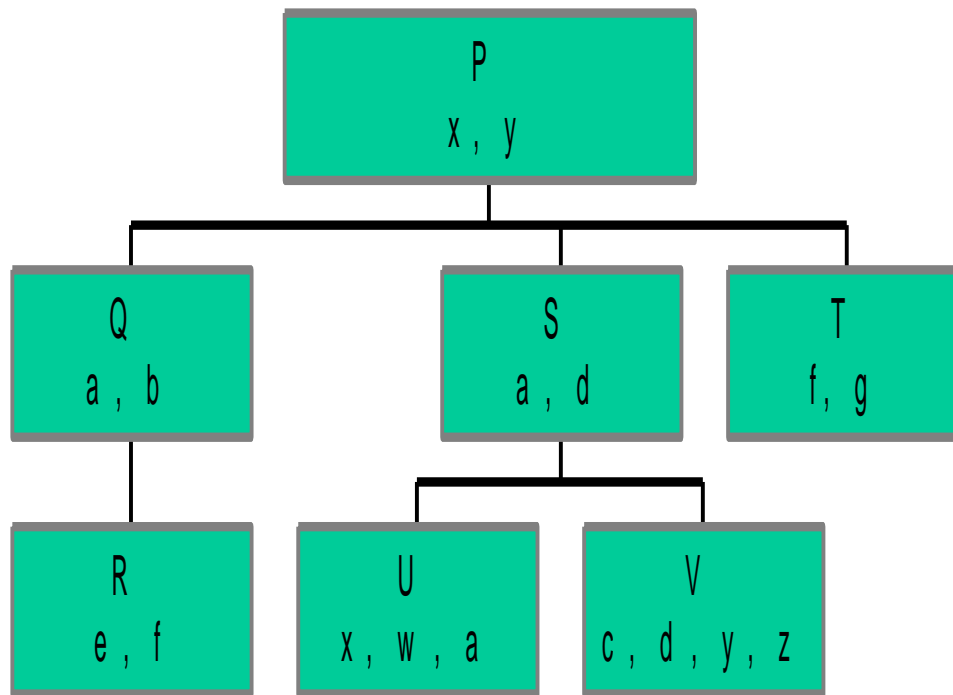
Regole di Visibilità

- Un identificatore è visibile nel programma o sottoprogramma in cui è dichiarato e in tutti i sottoprogrammi locali ad esso nei quali non è stato ridichiarato.
- Tutte le risorse di un programma o sottoprogramma devono essere dichiarate prima di essere usate
- Una risorsa globale è visibile ovvero accessibile ovvero usabile sempre e da tutti (main e sottoprogrammi) a meno che non venga oscurata (shadowing)
- Una risorsa locale ad un sottoprogramma P è visibile solo dalle istruzioni di P e dagli eventuali sottoprogrammi definiti in P

Vista di un Sottoprogramma

Esempio

Programma P



Vista di un Sottoprogramma

- Delimitazione spaziale di una risorsa
 - Le regole di visibilità degli identificatori stabiliscono l'*ambito* o *campo di visibilità* o *scopo degli identificatori* ovvero la zona di programma in cui è possibile fare riferimento a quell'identificatore.
- Delimitazione temporale di una risorsa
 - Le regole di visibilità definiscono anche la durata o il tempo di vita di una variabile ovvero l'intervallo di tempo in cui una variabile esiste (è allocata una area della RAM per essa)

Attributi delle variabili

- VARIABILE
 - NOME
 - VALORE
 - INDIRIZZO
 - TIPO
 - AMBITO
 - DURATA