Programmazione Procedurale

Sottoprogrammi
Passaggio dei Parametri
Parte a

versione 4.2

1

Sommario

- Semantica del Passaggio dei Parametri
- Simulazione
- Errori Frequenti
- Rapporto tra Funzioni e Procedure

2

Semantica

- Parametro
  - ⇨spazio nella memoria
- Memoria per un modulo di programma
  - ⇨memoria per le costanti locali (già discussa)
  - ⇨memoria per le variabili locali (già discussa)
  - ⇨memoria per i parametri
- Memoria relativa ai parametri
  - ⇨gestita in modo diverso dalle variabili

3

Semantica

ATTENZIONE
alla semantica operazionale del passaggio dei parametri

- Semantica
  - ⇨al passaggio dei parametri, viene riservata la memoria per i parametri
  - ⇨ad ogni parametro standard viene riservato uno spazio corrispondente al suo tipo
  - ⇨ad ogni parametro per riferimento viene riservato uno spazio adatto a contenere un indirizzo della memoria

4

Semantica

- Semantica (continua)
  - ⇨a ciascun parametro viene associato l'argomento corrispondente nella chiamata
  - ⇨nello spazio di memoria di ciascun parametro standard viene copiato il valore dell'argomento corrispondente
  - ⇨nello spazio di memoria di ciascun parametro per riferimento viene copiato l'indirizzo di memoria dell'argomento corrispondente

5

Semantica

- Semantica (continua)
  - ⇨terminato il passaggio dei parametri, comincia l'esecuzione del corpo del sottoprogramma
  - ⇨viene riservata eventuale memoria per costanti e variabili locali
  - ⇨vengono eseguite le istruzioni
  - ⇨le operazioni sulle costanti e sulle variabili locali vengono effettuate in modo ordinario

6

Semantica

- Semantica (continua)
  - ⇨le operazioni effettuate sui parametri standard vengono effettuate accedendo allo spazio di memoria relativo
  - ⇨viceversa, per effettuare un'operazione sul parametro per riferimento:
    - ⇨viene prelevato l'indirizzo dell'argomento
    - ⇨l'operazione viene effettuata direttamente sullo spazio di memoria dell'argomento

7

Semantica

- Semantica (continua)
  - ⇨l'esecuzione del corpo termina non appena viene incontrata l'istruzione return
  - ⇨al termine dell'esecuzione la memoria assegnata al sottoprogramma viene completamente rilasciata e si perde traccia dei dati relativi (costanti, variabili e parametri)
  - ⇨ricomincia l'esecuzione del modulo chiamante

8

Simulazione

- Riassumendo
  - ⇨i parametri standard sono spazi di memoria che si comportano in modo molto simile alle variabili
  - ⇨i parametri per riferimento sono solo "intermediari" (servono a fare da tramite tra il processore e i corrispondenti argomenti)
  - ⇨in particolare, consentono di effettuare modifiche sugli argomenti

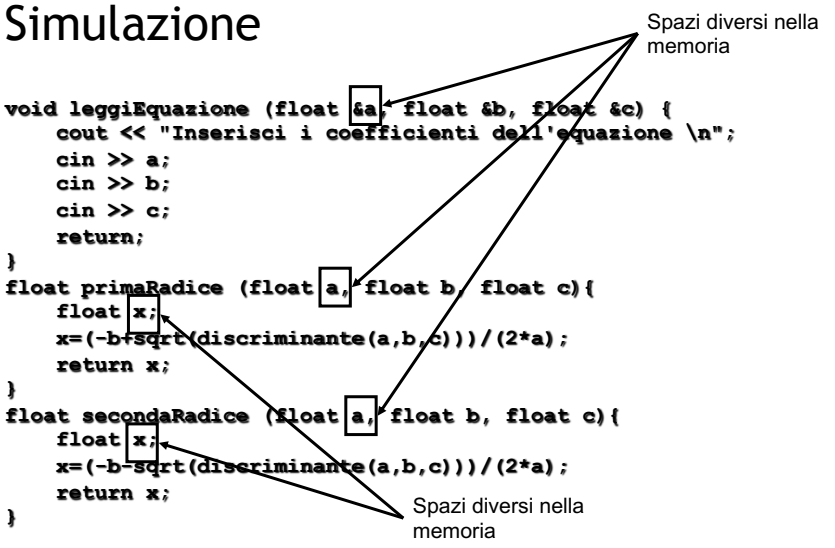
15

Simulazione

- Inoltre, sulla base della semantica descritta
  - ⇨dovrebbe essere chiaro che possono esserci dati diversi con lo stesso nome in moduli diversi
  - ⇨tra questi dati non c'è nessuna relazione, visto che corrispondono in generale a spazi diversi della memoria
  - ⇨l'unica forma di associazione tra dati diversi viene stabilita durante l'esecuzione dei sottoprogrammi relativi attraverso il passaggio dei parametri

16

Simulazione



17

Simulazione

- Esempio: il parametro “a”
  - ⇨ce ne sono diversi nei vari sottoprogrammi
  - ⇨il fatto che si chiamino allo stesso modo dipende dal fatto che nei vari sottoprogrammi rappresentano la stessa informazione (il primo coefficiente dell’equazione)
  - ⇨ma non devono necessariamente chiamarsi allo stesso modo (es: potrebbero chiamarsi w, y, v, ...)

18

Sottoprogrammi: Passaggio dei Parametri >> Simulazione

```
void leggiEquazione (float &w, float &b, float &c) {
    cout << "Inserisci i coefficienti dell'equazione \n";
    cin >> w;
    cin >> b;
    cin >> c;
    return;
}
float discriminante (float primoCoeff, float b, float c){
    float d;
    d=(-b+sqrt(discriminante(primoCoeff,b,c)))/(2*primoCoeff);
    return d;
}
float primaRadice (float v, float b, float c){
    float x;
    x=(-b + sqrt(discriminante(v, b, c))) / (2 * v);
    return x;
}
int main() {
    float a1, b1, c1;
    leggiEquazione(a1, b1, c1);
    if (discriminante(a1, b1, c1) >= 0 ) {...}
}
```

Questo codice è perfettamente corretto, anche se meno leggibile del precedente

19

Sottoprogrammi: Passaggio dei Parametri >> Errori Frequenti

Errori Frequenti

- Errori frequenti nell'utilizzo dei parametri
  - ⇨specificare un numero troppo alto di parametri per un sottoprogramma >> utilizzo di parametri inutili
  - ⇨omettere la & per parametri che devono essere per riferimento >> parametro standard al posto di parametro per riferimento

20

Errori Frequenti

- Sulla base di quanto detto
  - ⇨è opportuno utilizzare un parametro esclusivamente se è indispensabile
  - ⇨bisogna evitare parametri inutili
- Utilizzo di parametri inutili
  - ⇨complicano la lettura del programma
  - ⇨aumentano il livello di accoppiamento
  - ⇨facilmente introducono errori

21

Esempio di Parametro Inutile

```
float discriminante (float a, float b, float c
                     float d) {
   d = b*b-4*a*c;
   return d;
}

int main() {
   float a1, b1, c1;
   float a2, b2, c2;
   float d;
   ...
   if (discriminante(a1, b1, c1, d) >= 0 ...
```

d dovrebbe essere una variabile locale

22

Errori Frequenti

- Omettere la &
  - ⇨ATTENZIONE: grave errore frequente
  - ⇨perché omettere la & è un errore ?
  - ⇨perché il parametro viene considerato standard (per valore)
  - ⇨le modifiche vengono fatte direttamente sullo spazio di memoria del parametro
  - ⇨al termine la memoria viene rilasciata e si perde traccia delle modifiche

24

Errori Frequenti

```
le1.  void leggiEquazione
          (float a, float &b, float &c)
      {
le2.     cout << "Coefficienti ? ";
le3.     cin >> a;
le4.     cin >> b;
le5.     cin >> c;
le6.     return;
le7.  }

 m1.  int main() {
 m2.     float a1, b1, c1;
 m3.     float a2, b2, c2;
 m4.     float x1, y1, x2, y2;
 m5.     leggiEquazione(a1, b1, c1);
 m6.     leggiEquazione(a2, b2, c2);
 m7.     if (...)
```

#100	a1	-34256
#101	b1	5
#102	c1	2
#103	a2	xxx
#104	b2	xxx
#105	c2	xxx
#106	x1	xxx
#107	y1	xxx
#108	x2	xxx
#109	y2	xxx
#110		2
#111		#101
#112		#102

Coefficienti ? 2 5 2

25

Errori Frequenti

- Viceversa
  - ⇨utilizzare una & dove non serve è meno grave (non altera il funzionamento)
  - ⇨tutti i parametri potrebbero essere per riferimento (es: FORTRAN)
- Si tratta però di un errore logico
  - ⇨rende meno chiaro e leggibile il codice
  - ⇨consente di introdurre modifiche non desiderate al valore del parametro

Errori Frequenti

```
d1.  float discriminante
           (float& a, float& b, float& c)
     {
d2.    float d;
d3.    d = b * b - 4 * a * c;   // a = 5;
d4.    return d;
d5.  }

m1.  int main() {
...    ...
m7.    if (discriminante(a1,b1,c1)>=0
       && discriminante(a2,b2,c2)>=0){
m8.      x1=primaRadice(a1,b1,c1);
m9.      y1=secondaRadice(a1,b1,c1);
         ...
```

#100	a1	2
#101	b1	5
#102	c1	2
#103	a2	1
#104	b2	2
#105	c2	1
#106	x1	xxx
#107	y1	xxx
#108	x2	xxx
#109	y2	xxx
#110		#100
#111		#101
#112		#102
#113		9

e se per errore li sottoprogramma contenesse una istruzione di modifica ?

Rapporto tra Funzioni e Procedure

- Tecnicamente parlando
  - ⇨le funzioni non sono strettamente indispensabili
  - ⇨l'effetto di ogni funzione può essere simulata da una procedura equivalente
- Idea
  - ⇨sostituire il risultato della funzione con un parametro per riferimento

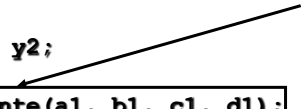
28

Esempio: discriminante

```
void calcolaDiscriminante
     (float a, float b, float c, float& d) {
   d = b*b-4*a*c;
   return;
}

int main() {
   float a1, b1, c1;
   float a2, b2, c2;
   float x1, y1, x2, y2;
   float d1, d2;
   calcolaDiscriminante(a1, b1, c1, d1);
   calcolaDiscriminante(a2, b2, c2, d2);
   if (d1 >= 0 && d2 >=0) {
      ...
```

al termine dell'esecuzione
il valore calcolato dal sottoprogramma
è contenuto nello spazio di memoria
della variabile d1



29

Rapporto tra Funzioni e Procedure

- Un ulteriore esempio
  - ⇨uno schermo che presenta all'utente un menu
  - ⇨ed acquisisce un comando numerico
- Due versioni equivalenti
  - ⇨`int schermoMenu()`
  - ⇨`void schermoMenu(int& scelta)`

30

Rapporto tra Funzioni e Procedure

```
int schermoMenu() {
   int scelta;
   cout << "1. Continua" << endl;
   cout << "2. Esci" << endl;
   cout << "Effettua la tua scelta" << endl;
   cin >> scelta;
   while ((scelta < 1) || (scelta > 2)) {
      cout << "Errore" << endl;
      cin >> scelta;
   }
   return scelta;
}

int main() {
   int sceltaUtente;
   sceltaUtente = schermoMenu();
   if (sceltaUtente == 1) {...}
}
```

la funzione restituisce un valore (1 oppure 2)
il programma principale attribuisce il valore allo spazio di memoria della variabile sceltaUtente

31

Rapporto tra Funzioni e Procedure

```
void schermoMenu(int& scelta) {
   cout << "1. Continua" << endl;
   cout << "2. Esci" << endl;
   cout << "Effettua la tua scelta" << endl;
   cin >> scelta;
   while ((scelta < 1) || (scelta > 2)) {
      cout << "Errore" << endl;
      cin >> scelta;
   }
   return;
}

int main() {
   int sceltaUtente;
   schermoMenu(sceltaUtente);
   if (sceltaUtente == 1) {...}
}
```

in questo caso la procedura modifica direttamente il valore dello spazio di memoria della variabile sceltaUtente

32

Rapporto tra Funzioni e Procedure

- Linea guida
  - ⇨è opportuno utilizzare una funzione ogni volta che il sottoprogramma ha come unica funzione quello di "calcolare" un valore singolo e restituirlo (es: discriminante) – maggiore leggibilità
  - ⇨se il sottoprogramma svolge anche altre operazioni (es: stampa del menu) è possibile scegliere l'una o l'altra versione

33

Rapporto tra Funzioni e Procedure

- In alcuni casi
  - ⇨è possibile utilizzare le procedure anche per calcoli specifici
- Esempio tipico
  - ⇨una funzione che deve restituire due valori contemporaneamente
  - ⇨la cosa più semplice è surrogarla con una procedura con due parametri per riferimento

34

Rapporto tra Funzioni e Procedure

- Esempio
  - ⇨calcolo delle radici reali di un'equazione
  - ⇨due possibili alternative
- Due funzioni

```
float primaRadiceReale(float a,float b,float c);
float secondaRadiceReale(float a,float b,float c)
```

- Una procedura con due param. per rif.

```
void calcolaRadici(float a, float b, float c
                   float &x1, float &x2);
```

35

Riassumendo

- Passaggio dei parametri
  - ⇨semantica operazionale delle chiamate
  - ⇨parametri standard
  - ⇨parametri per riferimento
- Errori Frequenti
  - ⇨ATTENZIONE

36

Esercizio

- Problema
  - ⇨calcolare il numero di pollici di un televisore, data la lunghezza e l'altezza dello schermo
- Definizione
  - ⇨il numero di pollici del televisore corrisponde alla misura in pollici della diagonale
  - ⇨1 pollice = 2.54 centimetri

37

Termini della Licenza

38