Programmazione Procedurale

Tipi Strutturati
Introduzione

versione 3.5

1

Sommario

- Panoramica
- Alcuni Esempi
  - ⇨Temperature Medie
  - ⇨Somma di Matrici
  - ⇨Partite di Campionato
- Nota sui Progetti

2

Panoramica

- Fino a questo punto
  - ⇨elementi di base + strutture di controllo: le funzionalità essenziali dei ling. procedurali
  - ⇨sottoprogrammi e programmazione modulare: miglioramento in termini di organizzazione del codice
- In queste lezioni
  - ⇨ulteriore miglioramento su rappresentazione dei dati attraverso i “tipi strutturati”

3

Panoramica

- In tutti i programmi
  - ⇨è necessario rappresentare attraverso le variabili i dati della realtà di interesse
- Finora
  - ⇨una variabile per ogni valore “semplice” (numero, carattere, stringa o booleano)
- In alcuni casi
  - ⇨questo meccanismo è di fatto inefficace

4

Panoramica

- In questa parte del corso
  - ⇨tipi strutturati
- Tipi strutturati
  - ⇨array e record
  - ⇨consentono di costruire strutture di variabili complesse
  - ⇨si tratta di una funzionalità fondamentale e delicata (richiede di ragionare sul modo di rappresentare i dati del problema nel progr.)

5

Alcuni Esempi

- Nel seguito
  - ⇨alcuni esempi che mettono in luce i limiti delle variabili dei tipi di base
  - ⇨motivazione per l'uso di tipi strutturati
- Esempi
  - ⇨Analisi delle Temperature a Potenza
  - ⇨Somma di Matrici
  - ⇨Segni delle Partite di Serie A

6

Temperature

- Esempio n.1: Temperature
  - ⇨date le temperature medie giornaliere del 2025 a Potenza, trovare (a) la temperatura media annuale; (b) il numero di temperature negative; (c) la temperatura minima; (d) la temperatura massima
- Dati del problema
  - ⇨dati di ingresso: 365 numeri reali; è necessario dichiarare 365 variabili

7

Temperature

- Una possibile soluzione

```
int main() {
 float t1;
 float t2;
 ... // altre 362 dichiarazioni
 float t365;
 ...
}
```

8

Temperature

- Difetti di questa soluzione
  - ⇨per la lettura dei dati sarebbe necessario scrivere una procedura con 365 parametri
  - ⇨nel corpo, sarebbe necessario scrivere una istruzione di lettura per ciascuna variabile
  - ⇨per il massimo e il minimo sarebbe necessario scrivere funzioni con 365 par.
  - ⇨la ricerca del massimo dovrebbe essere fatta caso per caso

9

Temperature

- Procedura per la lettura dei dati

```
void leggi (float &t1, ..., float &t365){
    cout << "Immetti i dati" << endl;
    cin >> t1;
    cin >> t2;
    ... // altre 362 istruzioni
    cin >> t365;
    return;
}
```

10

Temperature

- Funzione per il calcolo del minimo

```
float minimo (float t1, ..., float t365){
   float minimo;
   if (t1 <= t2 && t1 <= t3 && ... && t1 <= t365)
       minimo = t1;
   if (t2 <= t1 && t2 <= t3 && ... && t2 <= t365)
       minimo = t2;
   ... // altre 362 istruzioni if
   if (t365 <= t1 && t365 <= t2 && ... && t365 <= t364)
       minimo = t365;
   return minimo;
}
```

11

Temperature

- In effetti
  - ⇨teoricamente sarebbe possibile scrivere il codice usando le dichiarazioni viste finora
  - ⇨ma il codice sarebbe difficile da scrivere e di qualità molto scadente
  - ⇨moltissime istruzioni e alcune molto lunghe
- Causa del problema
  - ⇨il programma richiede di lavorare con una grossa collezione di dati dello stesso tipo

12

Temperature

- I tre problemi da risolvere
  - ⇨in questi casi, sarebbe utile poter dichiarare tutte le variabili con un'unica dichiarazione
  - ⇨poterle passare ai sottoprogrammi attraverso un unico parametro
  - ⇨poter scrivere le operazioni in modo compatto (per esempio usando i cicli)

13

Temperature

- La soluzione
  - ⇨gli array
- Array
  - ⇨funzionalità dei linguaggi di programmazione che consente di risolvere questi problemi
  - ⇨consentono di dichiarare con un'unica istruzione molte variabili dello stesso tipo
  - ⇨e di manipolarle in modo più efficace nelle istruzioni

14

Temperature

- Soluzione con array monodimensionale

```
const int N = 365;

int main() {
  float temperature[N];
  ...
}
```

Dichiarazione di array

Equivale a dichiarare N variabili di tipo float chiamate:
temperature[0]
temperature[1]
temperature[2]
...
temperature[N-1]

15

Temperature

- La soluzione ai tre problemi
  - ⇨gli array consentono di dichiarare molte variabili con un'unica istruzione
  - ⇨consentono di passare tutte le variabili ai sottoprogrammi attraverso un unico parametro (l'array nel suo complesso)
  - ⇨inoltre, il meccanismo di nomi consente di snellire le operazioni utilizzando i cicli (nomi uguali con un indice intero che cambia)

16

Temperature

- La soluzione con array
  - ⇨il progetto temperature contiene una soluzione del problema basata su array
  - ⇨è stata adottata una semplificazione
  - ⇨invece che temp. medie giornaliere, sono state considerate le temp. medie mensili
  - ⇨cambiando il valore della costante N, è possibile facilmente gestire medie giornaliere

>>letturaTemperature.cpp

17

Somma di Matrici

- Esempio n.2: Somma di Matrici
  - ⇨date due matrici di numeri di dimensioni NxM, calcolare e stampare la matrice somma
- Dati del problema
  - ⇨dati di ingresso: 2 x N x M numeri reali
  - ⇨dati di uscita: N x M numeri reali
  - ⇨anche in questo caso, se N ed M sono alti, il numero di dati da gestire è alto

18

Somma di Matrici

32	54	98
21	67	43
17	81	79

+

92	26	75
48	60	15
84	33	51

=

124	80	173
69	127	58
101	114	130

19

Somma di Matrici

- In linea di principio
  - ⇨sarebbe possibile risolvere il problema dichiarando 3 array di N x M variabili float

```
const int N = 10;
const int M = 12;
int main() {
   float array1[N * M];
   float array2[N * M];
   float array3[N * M];
   ...
}
```

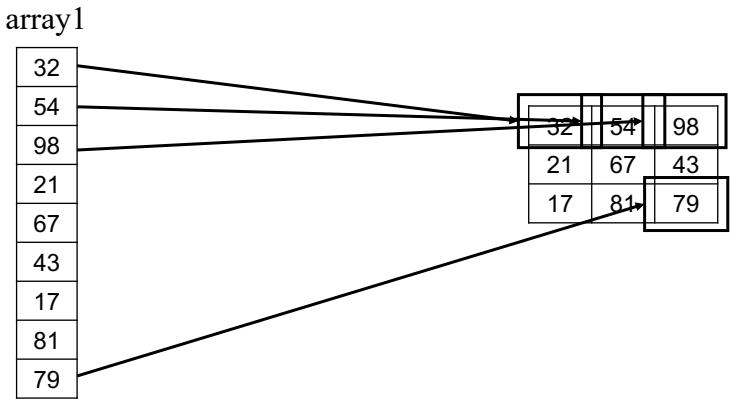
20

Somma di Matrici

- In questo modo
  - ⇨l'elemento n. 0 di array1 rappresenta l'elemento (1, 1) della prima matrice
  - ⇨l'elemento n. 1 di array1 rappresenta l'elemento (1, 2) della prima matrice
  - ⇨l'elemento n. 2 di array1 rappresenta l'elemento (1, 3) della prima matrice
  - ⇨l'elemento n. (N*M)-1 di array1 rappr. l'elemento (N, M) della prima matrice

21

Somma di Matrici



22

Somma di Matrici

- In effetti, però
  - in questo caso, a differenza dell'Esempio 1, i dati hanno caratteristiche "bidimensionali"
  - sono organizzati per righe e colonne
  - per semplificare il programma, sarebbe utile mantenere queste caratteristiche
  - e cioè poter identificare ciascuna variabile sulla base dell'indice di riga e di colonna dell'elemento corrispondente

23

Somma di Matrici

- Soluzione
  - un array bidimensionale
- Array multidimensionali
  - gli array possono avere molte dimensioni
  - gli array ordinari hanno un'unica dimensione (collezione lineare di variabili)
  - gli array bidimensionali hanno due dimensioni (concettualmente: riga e colonna)
  - possono esistere array con più di due dim.

24

Somma di Matrici

- Soluzione con gli array bidimensionali

```
const int N = 2;
const int M = 3;

int main() {
  float matrice1[N][M];
  float matrice2[N][M];
  float matrice3[N][M];
```

Dichiarazione di array bidimensionale

Equivale a dichiarare N*M variabili di tipo float:
matrice1[0][0]
matrice1[0][1]
...
matrice1[0][M-1]
matrice1[1][0]
matrice1[1][1]
...
matrice1[N-1][M-1]

25

Somma di Matrici

- In questo modo
  - ⇨si semplifica la scrittura dei cicli relativi agli array
  - ⇨in particolare, i cicli possono essere organizzati pensando alle righe ed alle colonne
  - ⇨tipicamente: due cicli nidificati, uno esterno per le righe e uno interno per le colonne

>> sommaMatrici1.cpp

26

Alcuni Esempi

- Riassumendo
  - ⇨gli array (mono e bidimensionali) sono una soluzione per rappresentare molti dati tutti dello stesso tipo
- In altri casi però
  - ⇨è necessario rappresentare molti dati, che richiedono più valori per essere rappresentati
  - ⇨questi valori sono spesso di tipi diversi tra loro

27

Partite di Serie A

- Esempio n. 3: Partite di Serie A
  - ⇨dati i risultati delle partite di campionato di serie A (squadre e gol), calcolare e stampare la colonna dei segni del totocalcio
- Dati del problema
  - ⇨una collezione di partite (suggerisce di usare un array di partite)
  - ⇨ma una partita non è un dato semplice
  - ⇨è fatta di due stringhe e due numeri interi

28

Partite di Serie A

- E' un caso frequente
  - ⇨spesso nei programmi bisogna rappresentare oggetti del mondo reale
  - ⇨attraverso varie delle loro caratteristiche
- Esempio
  - ⇨studente universitario (matricola, nome, cognome, data di nascita, anno di corso)
  - ⇨automobile (targa, modello, cilindrata)

29

Partite di Serie A

- Una possibile soluzione
  - ⇨rappresentare la collezione con vari array

```
const int N = 9;
int main() {
  string squadraCasa[N];
  string squadraTrasf[N];
  int goalCasa[N];
  int goalTrasferta[N];
  ...
```

La collezione di partite è rappresentata con tutti e 4 gli array

30

Partite di Serie A

31

Partite di Serie A

- In effetti
  - ⇨in alcuni linguaggi più datati (es: FORTRAN 77) è necessario fare così
  - ⇨nei linguaggi moderni, invece: record
- Record (o “Strutture”)
  - ⇨consentono di dichiarare gruppi di variabili, anche di tipi diversi per rappresentare oggetti della realtà

32

Partite di Serie A

- Dichiarazione di record
  - ⇨strategia a due passi
- Passo 1
  - ⇨descrivo la struttura del record (il "modello") al processore, dandogli un nome
- Passo 2
  - ⇨utilizzo il modello per dichiarare le variabili

33

Partite di Serie A

- Soluzione con record

```
struct partita {
  string squadraCasa, squadraTrasferta;
  int goalCasa, goalTrasferta;
  char segno;
};
const int N = 9;
int main() {
  partita p;
  partita partite[N];
  ...
```

Descrizione del "modello" della struttura partita

Dichiarazione di variabili secondo il modello della struttura partita

34

Partite di Serie A

- Semantica della dichiarazione

```
struct partita {
    string squadraCasa;
    string squadraTrasferta;
    int goalCasa;
    int goalTrasferta;
    char segno;
};

partita p;
```

Equivale a dichiarare
5 variabili:

```
string   p.squadraCasa;
string   p.squadraTrasferta
int      p.goalCasa;
int      p.goalTrasferta;
char     p.segno;
```

35

Partite di Serie A

- Con gli array

```
struct partita {
    string squadraCasa;
    string squadraTrasferta;
    int goalCasa;
    int goalTrasferta;
    char segno;
};

partita partite[N];
```

Equivale a dichiarare
N variabili di tipo partita:
partite[0]
partite[1]
partite[2]
...
partite[N-1]

Ognuna delle variabili corrisponde a sua volta a 5 variabili:

```
string   partite[1].squadraCasa;
string   partite[1].squadraTrasferta
int      partite[1].goalCasa;
int      partite[1].goalTrasferta;
char     partite[1].segno;
```

In totale: 5*N variabili

36

Partite di Serie A

- Modello di un record
  - ⇨è a tutti gli effetti un “nuovo tipo di dato”, con un identificatore proprio
  - ⇨a valori non semplici (non “atomici”), ma composti di più valori
  - ⇨è possibile utilizzarlo come si utilizzano i tipi di base
  - ⇨esempio: dichiarare variabili
  - ⇨esempio: dichiarare array di variabili

>> partite1.cpp

37

Nota sui Progetti

- Una caratteristica importante
  - ⇨nei vari progetti vengono manipolate collezioni di dati (temperature, numeri interi, matrici)
  - ⇨si suppone di conoscere esattamente la dimensione delle collezioni
  - ⇨quindi le componenti dei vari array sono sempre completamente utilizzate

38

Nota sui Progetti

ATTENZIONE
all'ipotesi sull'utilizzo delle componenti degli array

- In generale non è così
  - ⇨le collezioni possono avere dimensioni non fissate a priori
  - ⇨e variabili durante l'esecuzione del programma
  - ⇨ma questo è oggetto del modulo successivo
- In questo modulo
  - ⇨supporremo sempre di utilizzare tutte le componenti degli array

39

Riassumendo

- Tipi strutturati
  - ⇨semplificano la dichiarazione dei dati di un programma
- Classificazione
  - ⇨array monodimensionali
  - ⇨array bidimensionali
  - ⇨record

40

Termini della Licenza

41