Programmazione Procedurale

Tipi Strutturati
Array Monodimensionali

versione 3.5

1

Sommario

- Introduzione
- Dichiarazione di Array
  - ⇨Sintassi e Semantica
- Utilizzo degli Array
  - ⇨Passaggio dei Parametri per gli Array
  - ⇨Utilizzo delle Componenti
  - ⇨Scansione di un Array
- Utilizzo Corretto degli Array
- Array e Stringhe

2

Introduzione

- Array
  - ⇨funzionalità per il trattamento di collezioni di dati tutti dello stesso tipo
- Tre caratteristiche principali
  - ⇨dichiarazione di molte variabili con un'unica istruzione
  - ⇨passaggio dei parametri
  - ⇨utilizzo semplificato delle variabili nel programma attraverso cicli

3

Introduzione

- Idea
  - ⇨molte variabili dello stesso tipo
  - ⇨nome costruito da un identificatore comune
  - ⇨ed una serie di indici interi
- Dimensioni dell'array
  - ⇨array monodimensionali: un unico indice
  - ⇨array bidimensionali: due indici
  - ⇨array multidimensionali: tre o più indici

4

Dichiarazione di Array

- Alcuni esempi

```
char segniSchedina [13];
```
tipo delle variabili
identificatore dell'array
dimensione

```
const int N=12;
float temperature [N];
```
tipo delle variabili
identificatore dell'array
dimensione (costante simbolica)

```
const int N=100;
const int M=200;
string strano [N+M];
```
tipo delle variabili
identificatore dell'array
dimensione

5

Dichiarazione di Array

ATTENZIONE
valore costante intero e positivo per la dimensione

- Sintassi

```
<tipo> <nomeArray> [<dim>];
```

- Dove
  - ⇨<tipo> è uno dei tipi di dato del linguaggio
  - ⇨<nomeArray> è un identificatore
  - ⇨<dim> è un valore costante intero positivo (costante numerica o simbolica o espressione costante, ovvero espressione in cui gli operandi sono costanti)

6

Dichiarazione di Array

- Semantica
  - la dichiarazione ha l'effetto di dichiarare <dim> variabili del tipo specificato
  - le variabili sono dette "componenti" dell'array
  - alle var. corrisp. spazi contigui nella memoria
  - nomi delle variabili costituiti di due parti:
    - l'identificatore dell'array <nomeArray>
    - un indice intero che va da 0 a <dim>-1
    - ovvero: <nomeArray>[<indice>]

7

Dichiarazione di Array

- Esempio

```
char segniSchedina [13];
```

Ha l'effetto di dichiarare
13 variabili di tipo char:
segniSchedina[0]
segniSchedina[1]
segniSchedina[2]
...
segniSchedina[11]
segniSchedina[12]

#100	segniSchedina[0]	xxx
#101	segniSchedina[1]	xxx
#102	segniSchedina[2]	xxx
#103	segniSchedina[3]	xxx
#104	segniSchedina[4]	xxx
#105	segniSchedina[5]	xxx
#106	segniSchedina[6]	xxx
#107	segniSchedina[7]	xxx
#108	segniSchedina[8]	xxx
#109	segniSchedina[9]	xxx
#110	segniSchedina[10]	xxx
#111	segniSchedina[11]	xxx
#112	segniSchedina[12]	xxx

8

Dichiarazione di Array

- Esempio

```
const int N=12;
float temperature[N];
```

Ha l'effetto di dichiarare
12 variabili di tipo float:
temperature[0]
temperature[1]
temperature[2]
...
temperature[10]
temperature[11]

#1000	temperature[0]	xxx
#1001	temperature[1]	xxx
#1002	temperature[2]	xxx
#1003	temperature[3]	xxx
#1004	temperature[4]	xxx
#1005	temperature[5]	xxx
#1006	temperature[6]	xxx
#1007	temperature[7]	xxx
#1008	temperature[8]	xxx
#1009	temperature[9]	xxx
#1010	temperature[10]	xxx
#1011	temperature[11]	xxx

9

Dichiarazione di Array

- Attenzione
  - ⇨ la dimensione dell'array deve essere un valore costante
  - ⇨ NON può essere utilizzata una variabile oppure un'espressione generica
  - ⇨ il primo elemento ha sempre indice 0
  - ⇨ l'ultimo elemento ha sempre indice pari alla dimensione – 1 ("indicizzazione con base 0")
  - ⇨ es: temperature[4] è il quinto elemento
  - ⇨ temperature[12] NON esiste

10

Dichiarazione di Array

ATTENZIONE alla variabile array

- Semantica, un dettaglio importante
  - ⇨oltre alle componenti, la dichiarazione dell'array ha l'effetto di dichiarare una ulteriore variabile
  - ⇨l'array nel suo complesso
- Variabile array
  - ⇨il suo valore è l'indirizzo della memoria della prima componente dell'array

11

Dichiarazione di Array

- Esempio

```
char segniSchedina [13];
```

Ha l'effetto di dichiarare 14 variabili, 13 di tipo char e una che rappresenta l'array nel suo complesso

	segniSchedina	#100
#100	segniSchedina[0]	xxx
#101	segniSchedina[1]	xxx
#102	segniSchedina[2]	xxx
#103	segniSchedina[3]	xxx
#104	segniSchedina[4]	xxx
#105	segniSchedina[5]	xxx
#106	segniSchedina[6]	xxx
#107	segniSchedina[7]	xxx
#108	segniSchedina[8]	xxx
#109	segniSchedina[9]	xxx
#110	segniSchedina[10]	xxx
#111	segniSchedina[11]	xxx
#112	segniSchedina[12]	xxx

12

Dichiarazione di Array

- Esempio

```
const int N=12;
float temperature[N];
```

Ha l'effetto di dichiarare 13 variabili, di cui 12 di tipo float ed una che rappresenta l'array nel suo complesso

	temperature	#1000
#1000	temperature[0]	xxx
#1001	temperature[1]	xxx
#1002	temperature[2]	xxx
#1003	temperature[3]	xxx
#1004	temperature[4]	xxx
#1005	temperature[5]	xxx
#1006	temperature[6]	xxx
#1007	temperature[7]	xxx
#1008	temperature[8]	xxx
#1009	temperature[9]	xxx
#1010	temperature[10]	xxx
#1011	temperature[11]	xxx

Dichiarazione di Array

- Variabile array
  - ⇨è una variabile particolare
  - ⇨è teoricamente possibile stamparne il valore
  - ⇨ma non è possibile cambiarlo
- Utilizzo
  - ⇨viene utilizzata esclusivamente per il passaggio dei parametri (>>)
  - ⇨non deve essere utilizzata in altre istruzioni

Dichiarazione di Array

- Inizializzazione
  - ⇨come per le altre variabili, contestualmente alla dichiarazione è possibile inizializzare le componenti dell'array
- Sintassi
  - ⇨`<tipo> <nome> [<dim>]={<listaValori>};`
- Dove
  - ⇨<listaValori> è un elenco di valori del tipo delle compon. dell'array, separati da virgole

15

Dichiarazione di Array

- Esempio

```
int superEnalotto[6] = {1,24,23,5,82,4};
char segniSchedina[13] = {'1', '2', 'X',
  '1', 'X', '1', 'X', '1',  'X', '2',
  '1', '1', '2', '1', 'X'};
string mesi[12] = {"gennaio",
  "febbraio", "marzo", "aprile",
  "maggio", "giugno", "luglio", "agosto",
  "settembre", "ottobre", "novembre",
  "dicembre"};
```

16

Dichiarazione di Array

- Semantica
  - la memoria viene assegnata in modo ordinario alle componenti
  - inoltre, ciascuna componente viene inizializzata con un valore dell'elenco
  - l'associazione tra componente e valore viene fatta ordinatamente
  - questo modo di inizializzazione può essere usato solo una volta (alla dichiarazione)

17

Utilizzo degli Array

ATTENZIONE alla differenza tra dichiarazione e utilizzo

- Regola generale
  - le operazioni sull'array si fanno componente per componente
- Unica eccezione
  - passaggio dei parametri
  - è l'unica operazione che viene effettuata con la variabile che rappresenta l'intero array
  - la semantica deroga leggermente rispetto a quella ordinaria

18

Passaggio dei Parametri

ATTENZIONE
al passaggio dei parametri per gli array

- Nel sottoprogramma
  - ⇨viene specificato un parametro di tipo array
  - ⇨il parametro è sempre un parametro standard (non c'è la &)
- Nel modulo chiamante
  - ⇨viene usata come argomento la variabile che rappresenta l'intero array (variabile array)
  - ⇨il sottoprogramma acquisisce l'indirizzo di memoria della prima componente

19

Passaggio dei Parametri

costante globale

parametri di tipo array (NOTA: non viene mai usata la &)

```
const int N = 12;

void leggi(float temperature[N]);
void stampa(float temperature[N]);
float calcolaMedia(float temperature[N]);
int contaNegativi(float temperature[N]);
int posizioneMax(float arrayDati[N]);
int posizioneMin(float arrayDati[N]);
void stampaRisultato(float temperature[N],
                     float media, int negative,
                     int posMax, int posMin);
...
```

20

Tipi Strutturati: Array >> Passaggio dei Parametri

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

    int scelta;
    string nomeFile;
    bool continua = true;
    while (continua) {
        schermoMenu(scelta);
        if (scelta == 0) {
            continua = false;
        } else if (scelta == 1) {
            leggi(temperature);
        } else if (scelta == 2) {
            leggiNomeFile(nomeFile);
            leggiDaFile(temperature, nomeFile);
        } else if (scelta == 3) {
            stampa(temperature);
        } else if (scelta == 4) {
            ...
        }
}
```

come argomento viene usata la variabile che rappr. l'intero array

21

Tipi Strutturati: Array >> Utilizzo degli Array

Passaggio dei Parametri

- Alcune particolarità del meccanismo
  - ⇨ad un parametro di tipo array viene sempre assegnato uno spazio adatto a contenere un indirizzo della memoria
  - ⇨gli array non vengono mai "copiati" componente per componente
  - ⇨si ottengono gli effetti del passaggio per riferimento nonostante il parametro sia standard

22

Passaggio dei Parametri

```
const int N = 12;

void leggi(float temperature[N]);
void stampa(float temperature[N]);

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

si ottiene l'effetto del parametro per rif.to

#998	**N**	**12**
#999	**temperature**	**#1000**
#1000	temperature[0]	xxx
#1001	temperature[1]	xxx
#1002	temperature[2]	xxx
#1003	temperature[3]	xxx
#1004	temperature[4]	xxx
#1005	temperature[5]	xxx
#1006	temperature[6]	xxx
#1007	temperature[7]	xxx
#1008	temperature[8]	xxx
#1009	temperature[9]	xxx
#1010	temperature[10]	xxx
#1011	temperature[11]	xxx
#1012	**temperature**	#1000

23

Passaggio dei Parametri

- Riassumendo
  - ⇨i parametri di tipo array sintatticamente sembrano sempre parametri standard
  - ⇨nonostante questo, il sottoprogramma è in grado di modificare le componenti dell'arg.
- Nota
  - ⇨specificare la & prima di un parametro di tipo array non solo è inutile, ma è anche un errore

24

Passaggio dei Parametri

- Una ulteriore annotazione
  - ⇨come detto i sottoprogrammi possono avere parametri che sono array
  - ⇨esclusivamente parametri standard
- Inoltre
  - ⇨le funzioni non possono restituire come risultato un array
  - ⇨ovvero il tipo del risultato di una funzione non può essere un array

25

Utilizzo delle Componenti

- Le restanti operazioni
  - ⇨vengono effettuate sulle singole componenti
- Componente dell'array
  - ⇨variabili ordinarie
  - ⇨tranne che per il nome particolare
  - ⇨la sintassi consente di manipolarle efficacemente attraverso l'uso di cicli

26

Utilizzo delle Componenti

- Utilizzo di una componente
  - ⇨come per tutte le altre variabili
- Esempio: primo segno della schedina

```
char segniSchedina[13];
cin >> segniSchedina[0];
cout << segniSchedina[0];
if (segniSchedina[0] == 'X') {
   cout << "Pareggio" << endl;
}
```

27

Utilizzo delle Componenti

ATTENZIONE al meccanismo per l'utilizzo delle componenti

- Però...
  - ⇨esiste anche un modo alternativo per usare le componenti, basato sul particolare nome
  - ⇨es: `segniSchedina[i]`
- Idea
  - ⇨viene specificata come indice un'espressione
  - ⇨il processore al momento dell'esecuzione calcola il valore dell'espressione e stabilisce quale componente utilizzare

28

Utilizzo delle Componenti

- Sintassi per l'uso delle componenti

  `<nomeArray>[<espressione>]`
- Dove
  - ⇨<espressione> è un'espressione a valori interi
- Esempi

  `temperature[3], segniSchedina[0]`

  `segniSchedina[i]`

  `prova[i + j]`

29

Utilizzo delle Componenti

- Semantica
  - ⇨quando il processore incontra un'istruzione in cui viene utilizzata una componente dell'array deve capire quale componente utilizzare (quale spazio nella memoria)
  - ⇨per farlo calcola il valore dell'espressione tra parentesi quadre
  - ⇨utilizza il valore calcolato come indice per identificare la componente da utilizzare

30

Scansione degli Array

- "Scansione (Visita) di un array"
  - ⇨operazione tipica sulle variabili dell'array
  - ⇨consiste nell'utilizzare un ciclo per poter eseguire la stessa operazione su tutte le componenti dell'array
- Vantaggio
  - ⇨è possibile effettuare le stesse operazioni su molte variabili scrivendo un'unica volta le istruzioni corrispondenti

31

Scansione degli Array

- Idea
  - ⇨si utilizza un ciclo for
  - ⇨con una variabile di controllo (es: i) che varia da 0 fino alla dimensione dell'array (es: N)
  - ⇨il corpo contiene una istruzione che agisce sulla componente i-esima dell'array (es: temperature[i])

32

Scansione degli Array

- Struttura tipica

```
const int N = 12;
float temperature[N];
int i;

for (i = 0; i < N; i++) {
   <operazione su temperature[i]>
}
```

es: cin >> temperature[i];
cout << temperature[i];
if (temperature[i] > 0) {... }

i = 0;
i = 1;
i = 2;
i = 3;
i = 10;
i = 11;
i = 12;

#998	**N**	**12**
#999	**temperature**	**#1000**
#1000	temperature[0]	xxx
#1001	temperature[1]	xxx
#1002	temperature[2]	xxx
#1003	temperature[3]	xxx
#1004	temperature[4]	xxx
#1005	temperature[5]	xxx
#1006	temperature[6]	xxx
#1007	temperature[7]	xxx
#1008	temperature[8]	xxx
#1009	temperature[9]	xxx
#1010	temperature[10]	xxx
#1011	temperature[11]	xxx

33

Esempio: Temperature

ciclo per la lettura viene eseguito per i da 0 ad N - 1

```
void leggi(float temperature[N]){
    cout << "Immetti le temperature \n";
    for (int i = 0; i < N; i++) {
        cout << "Mese n. " << i+1 << ": ";
        cin >> temperature[i];
    }
    return;
}
```

alla I esecuzione: i=0 - cin>> temperature[0]
alla II esecuzione: i=1 - cin>> temperature[1]
alla III esecuzione: i=2 - cin>> temperature[2]
alla IV esecuzione: i=3- cin>> temperature[3]
...

34

Esempio: Temperature

```
void stampa(float temperature[N]){
    cout << "Temperature Medie" << endl;
    cout << "----------------------------" << endl;
    for (int i = 0; i < N; i++) {
        cout << "Mese n. " << i+1 << ":\t";
        cout << temperature[i] << endl;
    }
    return;
}
```

ciclo per la stampa

35

Utilizzo delle Componenti

- Nota bene la differenza

```
const int N = 12;
float temperature[N];
int i;
cin >> i;
cin >> temperature[3];
cout << temperature[0];
```

istruzioni che utilizzano sempre lo stesso spazio nella memoria

```
const int N = 12;
float temperature[N];
int i;
cin >> i;
cin >> temperature[i];
for (i = 0; i < N; i++) {
    cout << temperature[i];
}
```

istruzioni che utilizzano spazi diversi nella memoria al variare di i

36

Utilizzo delle Componenti

- ATTENZIONE alle parentesi quadre
  - ⇨vengono utilizzate in due modi diversi
- Nella dichiarazione dell'array
  - ⇨per indicare la dimensione dell'array
  - ⇨es: `char segniSchedina[13];`
- Nell'utilizzo delle componenti
  - ⇨per specificare una singola componente
  - ⇨es: `cout << segniSchedina[5];`

37

Utilizzo delle Componenti

- Una postilla
  - ⇨è possibile usare espressioni arbitrarie
- Esempio
  - ⇨un programma per stampare le prime cinque componenti in posizione pari di un array

```
void stampaCinquePari (int prova[100]) {
  for (int i = 0; i < 5; i++)
    cout << prova[i * 2];
  return;
}
```

38

Utilizzo Corretto degli Array

- In alcuni linguaggi di programmazione
  - ogni volta che viene utilizzata una componente di un array il processore verifica che l'indice sia tra i valori ammessi
  - es: Pascal, Java
- In altri linguaggi
  - questa verifica non viene fatta
  - tra questi linguaggi ci sono il C, il C++ ed il FORTRAN

39

Utilizzo Corretto degli Array

- Di conseguenza, in questi linguaggi
  - è possibile commettere errori dovuti a "sconfinamento" al di fuori delle componenti di un array senza che il processore li segnali
- Esempio

```
int main() {
  int prova[10];
  prova[200] = 1;
}
```

grave errore logico:
viene cambiato il valore
di un registro della memoria
che non appartiene all'array
(in questo caso, tipicamente,
è un registro dell'array temp)

40

Utilizzo Corretto degli Array

- Comportamento tipico del programma
  - ⇨l'errore non viene segnalato dal compilatore
  - ⇨il programma viene eseguito senza che il processore rilevi errori
  - ⇨successivamente, però, la modifica sulla memoria scatena errori latenti
- In alcuni casi (fortunati)
  - ⇨viene immediatamente segnalato un errore perché la locazione toccata è riservata

41

Utilizzo Corretto degli Array

ATTENZIONE
ad evitare problemi di sconfinamento

- Storicamente
  - ⇨lo sconfinamento è una delle cause principali di errori nei linguaggi C/C++
  - ⇨questa è la ragione per cui è stato impedito in Java
- Evitare lo sconfinamento
  - ⇨è compito del programmatore
  - ⇨bisogna fare attenzione a questo tipo di errori

42

Array e Stringhe

- Il tipo string
  - ⇨sequenze di caratteri di lunghezza arbitraria
- In effetti
  - ⇨per certi versi una variabile di tipo string è assimilabile ad un array
  - ⇨in particolare un array di variabili di tipo char
  - ⇨i caratteri di una stringa sono manipolabili come variabili di tipo char

43

Array e Stringhe

- Esempio
  - ⇨due frammenti equivalenti

```
string s;
s = "prova";
cout << s;
```

```
string s;
s = "*****";
s[0] = 'p';
s[1] = 'r';
s[2] = 'o';
s[3] = 'v';
s[4] = 'a';
int i;
for (i = 0; i < s.size(); i++){
   cout << s[i];
}
```

NOTA: le stringhe si comportano in modo simile agli array ma NON sono array (sono oggetti)

44

Riassumendo

- Array
  - ⇨gestione di collezioni di dati dello stesso tipo
  - ⇨in questa lezione monodimensionali
- Dichiarazione
- Utilizzo
  - ⇨lettura e stampa dei valori
- Correttezza dell'utilizzo (ATTENZIONE)

45

Termini della Licenza

46