ARRAY

Lorenzo Porcelli

Introduzione

Un vettore, array, è una *struttura dati* caratterizzata da una sequenza di dati tutti dello stesso tipo e che occupano posizioni di memoria adiacenti.

Lo spazio occupato da un vettore non varia durante l'esecuzione del programma.

Il numero di celle, cioè il numero di dati utilizzabili, è definito al momento della dichiarazione.

Dichiarazione

La dichiarazione specifica:

- il tipo dei dati,
- il numero di celle a disposizione,
- il nome del vettore.

```
int c[12];
```

- c è il nome del vettore
- 12 sono le celle del vettore
- Ogni cella è di tipo int

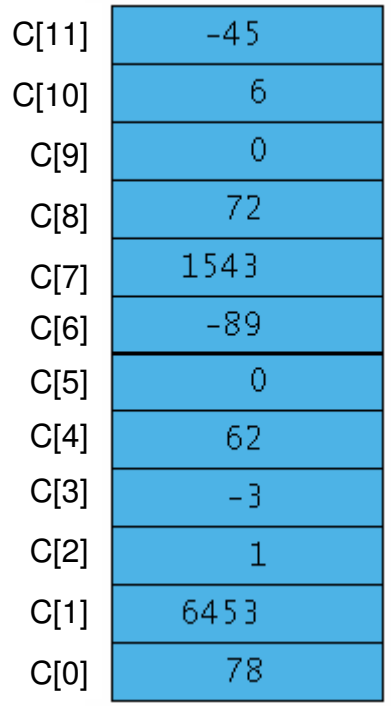
Accesso alle celle

Per accedere alla cella di un vettore bisogna specificare il nome del vettore, seguito dalla posizione dell'elemento racchiusa tra parentesi quadre.

Utilizzando la dichiarazione precedente:

```
printf( "%d", c[2] );
i=2; k=3;
c[i+k]=1;
c[3] = c[1] + c[2];
```

Le singole celle sono int, quindi su ogni singola cella sono permesse tutte le operazioni sugli int.



Dichiarazione con inizializzazione

Si può dichiarare un vettore e inizializzarlo.

```
float x[5] = { 1.0, 2.0, 1.3, 4.5, 0.3 };
int c[10] = { 1, 2, 3 };
```

Nel secondo caso le celle non inizializzate vengono poste a zero.

```
double y[ ] = { 1.0, 2.0, 3.0 };
```

y è un vettore di 3 elementi.

Utilizzo di un vettore

L'istruzione di ciclo for è ideale per le operazioni sui vettori.

```
int c[20], i, s=0;
.....
for(i=0; i<20; ++i)
   scanf("%d", &c[i]);

for(i=0; i<20; ++i)
   printf("%d", c[i]);

for(i=0; i<20; ++i)
   s += c[i];
```

Funzioni e vettori

```
float media( int v[ ], int dim ) {
        int i;
        float s=0.0;

        for(i=0; i<dim; ++i)
                s += v[i];
        return s / dim;
}

int main() {
        int c[ 100 ];
        float m;
        ……
        m = media( c, n );
        …..
        return 0;
}
```

- Nei parametri compare v[ ]; significa che v è un vettore.
- Il secondo parametro specifica quanti sono gli elementi validi di v, cioè gli elementi da utilizzare; la dimensione reale di v.
- Al momento della chiamata si indica il nome del vettore, e il numero di elementi validi.

Vettori e funzioni

Il passaggio di un vettore ad una funzione avviene sempre per riferimento: la funzione chiamata ha in ingresso l'indirizzo del vettore, cioè l'indirizzo della prima cella del vettore.

La funzione quindi opera direttamente sul vettore del chiamante.

Vettore come parametro

```
int leggi( float x[ ] ) {
    int n, i;

    scanf("%d", &n);
    for(i=0; i<n; ++i)
      scanf("%f", &x[i]);
    return n;
}

int main() {
    float vett[100];
    int dim;

    dim = leggi( vett );
    stampa( vett, dim );

    return 0;
}
```

La funzione leggi permette l'inserimento di n numeri in un vettore.

L'indirizzo del vettore è fornito al momento della chiamata.

La funzione restituisce la dimensione reale del vettore, cioè il numero di valori effettivamente letto.

Tale valore viene inviato alla funzione stampa, che procederà alla visualizzazione di tutti gli elementi del vettore.

vett == &vett[0]: il nome di un vettore è l'indirizzo della prima

printf("%p %p", vett, &vett[0]);

Occupazione della memoria

```
int main() {
    float vett[20];
    int i;
    for(i=0; i<20; ++i) {
        if(i%5 == 0) printf("\n");
        printf("%p ", &vett[i]);
    }

    return 0;
}
```

Le celle di un vettore occupano zone di memoria consecutive.

Gli indirizzi saranno diversi ad ogni esecuzione ma sono consecutivi e si ottengono incrementando l'indirizzo di partenza della dimensione della cella, nell'esempio 4 bytes.

```
90fb8 90fbc 90fc0 90fc4 90fc8
90fcc 90fd0 90fd4 90fd8 90fdc
90fe0 90fe4 90fe8 90fec 90ff0
90ff4 90ff8 90ffc 91000 91004
```

Vettore ed elemento

```
int pari( int n ){
    return  n%2 == 0;
}

int sommaPari(int v[], int n) {
    int s=0, i;

    for(i=0; i<n; ++i)
          if( pari(v[i]) ) s += v[i];
    return s;
}
```

La funzione sommaPari richiama la funzione pari alla quale invia un singolo elemento del vettore, cioè un int.

Vettori e puntatori

Ogni vettore viene passato per indirizzo alla funzione.

Una funzione che ha come parametro un vettore può essere dichiarata in due modi diversi.

```
int leggi( float *x ) {
    int n, i;

    scanf("%d", &n);
    for(i=0; i<n; ++i)
        scanf("%f", &x[i]);
    return n;
}
```

```
int leggi( float x[] ) {
    int n, i;

    scanf("%d", &n);
    for(i=0; i<n; ++i)
        scanf("%f", &x[i]);
    return n;
}
```

Aritmetica dei puntatori

Sia int a[100], i;

La scrittura

a[i]

è equivalente a

*(a+i)

Se p è un puntatore

p[i]

è equivalente a

*(p+i)

```
int leggi( int *x ) {
    int n, i;

    scanf("%d", &n);
    for(i=0; i<n; ++i)
        scanf("%d", x+i);

    return n;
}
```

Aritmetica dei puntatori

```
double a[2], *i, *f;

i = a;
f = a+1;

printf("%p %p\n", i, f);
printf("%d\n", f-i);
printf("%d\n", (int)f - (int)i);
```

Il risultato prodotto sarà:

```
90fa0 90fa8
1
8
```

Gli indirizzi variano ad ogni esecuzione ma differiscono sempre di 8 bytes.

Puntatori: attenzione!!!

```
void f(int *x){
    int i;

    for(i=0;i<3;++i)
        x[i]=i;
}

int main(){
    int a0=5, a1=5, a2=5, a3=5;
    int i, *p;

    f(&a0);
    printf("%d %d %d %d\n", a0, a1, a2, a3);
    p = &a0;
    for(i=0;i<=3; ++i)
        printf("%d ", *(p+i) );

    return 0;
}
```

La funzione f ha in ingresso un indirizzo.

Modifica 3 zone di memoria consecutive.

Le variabili a0, a1, a2 vengono così modificate.

Osservare anche come vengono stampate!!

Il risultato ottenuto è il seguente:

0 1 2 5

0 1 2 5

Vettori e funzioni: proposte di lavoro

Risolvere ogni esercizio realizzando una o più funzioni:

- Dato un vettore di interi calcolare la media aritmetica dei numeri pari in esso contenuti.
- Dato un vettore di interi copiare in un nuovo vettore tutti i numeri pari in esso contenuti.
- Dato un vettore di numeri interi cancellare tutti i numeri pari in esso contenuti.