UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Progettare un'interfaccia: costruttore con inizializzazione

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Agenda

- Proseguiamo lo sviluppo di vector
- Inizializzazione
  - initializer_list
- Alcuni casi di ambiguità

Richiamo – vector

- Riprendiamo il nostro vector

```
class vector {
    int sz;
    double* elem;

public:
    vector(int s = 0)
        : sz{s}, elem{new double[s]} { if(s == 0) elem =
nullptr; }
    ~vector()
        { delete[] elem; }
    // ... - include push_back(double d)
};
```

Inizializzazione

- Al momento possiamo solo inizializzare usando:

```
vector v3;              // lungo e ripetitivo
v3.push_back(1.2);
v3.push_back(7.89);
v3.push_back(12.34);
```

- Ora vogliamo poter inizializzare il vettore nel solito modo:

```
vector v1 = { 1.2, 7.89, 12.34 };    // più compatto!
```

Initializer list

- Una scrittura tipo:

```
vector v1 = { 1.2, 7.89, 12.34 };    // più compatto!
```

fornisce un oggetto della standard library di tipo initializer_list<T>

– Possiamo quindi aggiungere questo costruttore:

```
class vector {
// ...
    vector(initializer_list<double> lst)
        : sz{lst.size()}, elem{new double[sz]}
    {
        copy(lst.begin(), lst.end(), elem);
    }
// ...
};
```

passato per copia, non reference
(non ci sono problemi di efficienza)

Initializer list

```
class vector {
// ...
    vector(initializer_list<double> lst)
        : sz{lst.size()}, elem{new double[sz];}
    {
        copy(lst.begin(), lst.end(), elem);
    }
// ...
};
```

copy algorithm (standard library)

- Copy algorithm: copia una sequenza di elementi delimitata dai suoi primi due argomenti a una sequenza puntata dal terzo argomento
  - **Iteratori** per la definizione della sorgente

Initializer list

- Osserviamo un dettaglio:

```
vector(initializer_list<double> lst)
```

  - È passato per copia!
  - initializer_list è usato in questo modo, come richiesto dal linguaggio
  - initializer_list è un handle a elementi allocati "altrove"

Casi di ambiguità

- Le seguenti sono entrambe corrette

```
vector v1 {3};
vector v2(3);
```

- Qual è la differenza?

```
vector v1 {3};
vector v2(3);
```

Il primo inizializza il vettore a un elemento di valore 3, il secondo crea un vettore di tre elementi.

- Qual è la differenza?

```
vector v11 = {1, 2, 3};
vector v12 {1, 2, 3};
```

Sono perfettamente equivalenti, cambia solo la scrittura.

Casi di ambiguità

- Le seguenti sono entrambe corrette

```
vector v1 {3};
vector v2(3);
```

- Qual è la differenza?

```
vector v1 {3}; // un elemento inizializzato a 3
vector v2(3);  // tre elementi inizializzati a 0.0
```

- Qual è la differenza?

```
vector v11 = {1, 2, 3};
vector v12 {1, 2, 3};
```

Casi di ambiguità

- Le seguenti sono entrambe corrette

```
vector v1 {3};
vector v2(3);
```

- Qual è la differenza?

```
vector v1 {3}; // un elemento inizializzato a 3
vector v2(3);  // tre elementi inizializzati a 0.0
```

- Qual è la differenza?

```
vector v11 = {1, 2, 3};
vector v12 {1, 2, 3};          // equivalenti
```

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Progettare un'interfaccia: costruttore con inizializzazione

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE