Costruttori di Copia

Costruttore di copia

- Esiste un tipo speciale di costruttore detto costruttore di copia
- Un costruttore di copia serve per inizializzare un oggetto tramite un altro oggetto
- Il costruttore di copia è una funzione membro che ha lo stesso nome della classe e ha come argomento un riferimento costante ad un oggetto della stessa classe

```
NomeClasse(const NomeClasse &)
```

- Il compilatore crea un costruttore di copia bit a bit di default, cioe' ogni dato membro viene copiato

Quando viene invocato il costruttore di copia

- Ci sono tre casi in cui viene invocato il costruttore di copia:
  - inizializzazione esplicita
  - passaggio per valore ad una funzione
  - restituzione di un oggetto temporaneo

Inizializzazione Esplicita

```
Time t1(12,00);
Time t2(t1);
Time t3=t1;
```

- Le due forme **`t2(t1)`** e **`t3=t1`** sono equivalenti
- Nota: **`t3=t1`** non è una assegnazione ma una inizializzazione
- Ogni dato membro di t2 viene inizializzato con i valori dei dati membro corrispondenti di t1

Passaggio per valore ad una funzione

```
void func(Time);

void main(){
   Time t(12,00);
   func(t);
}
void func(Time myT){
   cout<<myT;
}
```

- Ogni dato membro di *myT* viene inizializzato con i valori dei dati membro corrispondenti di *t*
- Nota: utilizzando il costruttore standard, *myT* sarebbe inizializzato con valori (di default) diversi da quelli di *t* e questo sarebbe un errore logico

Restituzione di un oggetto temporaneo

```
Time func(Time);

void main(){
   Time t(12,00), t2;
   t2=func(t);
}

Time func(Time myT){
   myT.ora+=1;
   return myT;
}
```

Spiegazione

- L'oggetto *myT* è locale alla funzione func.
- Non è pertanto possibile riferirsi a tale oggetto al di fuori della funzione.
- Quando viene eseguita l'istruzione *t2=func(t)* accade in realtà la seguente cosa:
  - viene creato un oggetto temporaneo [result]
  - questo oggetto viene inizializzato per copia con l'oggetto restituito da *func:* [result]=myT di func(t)
  - viene distrutto myT
  - viene eseguita l'assegnazione t2=[result]
  - viene distrutto [result]

Nota sul Costruttore di copia

- Si deve necessariamente passare un riferimento per alias al costrutture di copia
- Non sarebbe logicamente possibile effettuare un passaggio di parametri per valore
- Dare la possibilità di creare un costruttore di copia con passaggio di parametro per valore comporta una dipendenza logica circolare
  - infatti nella funzione costruttore di copia il parametro passato dovrebbe essere copiato
  - ma questo richiamerebbe proprio il costruttore di copia (cioe' la stessa funzione)
  - il ciclo continuerebbe senza fine

Costruttore di Copia con dati dinamici

- E' necessario creare esplicitamente un costruttore di copia quando un oggetto ha dati membro allocati dinamicamente
- Per oggetti con dati allocati dinamicamente la copia bit-a-bit copia solo il dato puntatore, cioè l'indirizzo di memoria
- Quando un oggetto B è inizializzato come copia di un oggetto A si ha che entrambi gli oggetti hanno un puntatore che indirizza la stessa area di memoria
- Quando si dealloca il dato per un oggetto questo non esiste più nemmeno per la copia
- Si corre il rischio di deallocare due volte la stessa area di memoria!

Costruttore di copia

```
class Array{
public:
   Array(int usr_size=10){
        size=usr_size;
        ptr=new int[size];
   }
   Array(const Array& copia){
        size=copia.size;
        ptr=new int[size];
        for(int i=0;i<size;i++)
              ptr[i]=copia.ptr[i];
   }
   ~Array(){delete [] ptr;}
private:
   int *ptr;
   int size;
};
```