UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

3. Initialization List in Cpp

2 esercizi risolti + 1 extra

Tutorato di
Programmazione Avanzata

RELATORE
Imberti Federico

SEDE
Dalmine, BG

Key-points della teoria: Cpp

1. Cpp è un linguaggio orientato agli **oggetti** che nasce come evoluzione di C, il quale è invece procedurale. Introduce:
    a. Oggetti, con il concetto di incapsulamento:
        i. Information hiding;
        ii. Definizione di un'interfaccia;
        iii. Controllo sulla modifica dei dati…
    b. Maggiore espressività nel modellare il mondo.
2. Disponiamo sia di **reference** (&) che di **puntatori** (*), una reference è un puntatore che non può essere NULL (evita di dover fare dei controlli in una funzione per verificare che il parametro passato abbia un valore!)

Key-points della teoria: La memoria

La memoria a disposizione del programmatore è suddivisa in 2 macro-aree:

a. Memoria del codice, contenente il codice sorgente da eseguire

b. Memoria dei **dati**, la quale si evolve con il programma, divisa in:

   Stack (LiFo): dimensione fissa, contiene variabili, parametri e indirizzi di ritorno

```
void example() {
    int x = 10;  // 'x' is stored in stack memory
}  // 'x' is deallocated automatically when the function ends
```

```
{ //Imagine you are in the main method...
    int x = 10;  // 'x' is stored in stack memory
}
// 'x' is deallocated automatically when we leave the block
```

Key-points della teoria: La memoria

La memoria a disposizione del programmatore è suddivisa in 2 macro-aree:

a. Memoria del codice, contenente il codice sorgente da eseguire

b. Memoria dei **dati**, la quale si evolve con il programma, divisa in:

   Heap (FiFo): usata per l'allocazione dinamica dei dati; gestita dal programmatore; usata per contenere strutture dati di grandi dimensioni (oggetti, array, …)

```
void example() {
    int* ptr = new int(10);  // Memory for the integer is allocated on the heap
    delete ptr;  // The programmer is responsible for freeing the memory
}
```

Key-points della teoria: Initializer list

1. Permettono di eseguire delle operazioni prima che il codice nel costruttore venga eseguito;
2. Usate per inizializzare **attributi** della classe prima che questa sia creata;
3. L'ordine di inizializzazione degli attributi segue l'ordine in cui sono stati definiti e non quello della initializer list.

```
class PinoPasticcino {
     Public:
          int a;          // Variabile sullo stack
          int& b;         // Reference
          const int c;    // Costante

          PinoPasticcino(int x, int& y, int z) : a(x), b(y), c(z) {
            // Constructor body
          }
};
```

Key-points della teoria: Initializer list

1. Talvolta sono **necessarie**:
   a. Vogliamo inizializzare un attributo sullo stack che non dispone di un **costruttore di default**;
   b. Dobbiamo inizializzare un attributo che è una **reference**;
   c. Stiamo costruendo una **sottoclasse** e vogliamo usare un costruttore della superclasse che non sia quello di default;
   d. Quando stiamo inizializzando dei **const members**.
2. Talvolta sono **facoltative**:
   a. Vogliamo inizializzare un attributo sullo stack avente un costruttore di default;
   b. Dobbiamo inizializzare un attributo che è un puntatore…

UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

Domandine?

Esercizio 1/2

Definire una classe Motore con un attributo intero che rappresenta la Cilindrata. Definire una classe Automobile con un attributo di tipo Motore. Sviluppare tre implementazioni diverse della classe Automobile in cui l'attributo Motore è rispettivamente un (a) puntatore a una porzione di memoria nello stack, (b) puntatore a una porzione di memoria nell'Heap come puntatore, (c) reference a una porzione di memoria nell'Heap. Per ciascuno di questi tre casi discutere se (a) è possibile utilizzare l'Inizializer list e (b) è possibile inizializzarlo all'interno del costruttore.

Esercizio 2/2

Definire una classe Partita che rappresenta una partita di EURO 2024. La classe partita ha due attributi di classe Nazionale che rappresentano le nazionali di calcio che giocano la partita. Definire una classe Nazionale che ha come attributo il nome del relativo paese. Sviluppare tre implementazioni diverse della classe Partita in cui i suoi attributi sono rispettivamente (a) due puntatori a una porzione di memoria nello stack, (b) due puntatori a una porzione di memoria nell'Heap, (c) due riferimenti a una porzione di memoria nell'Heap. Per ciascuno di questi tre casi discutere se (a) è possibile utilizzare l'inizializer list e (b) è possibile inizializzarlo all'interno del costruttore.

Esercizio Extra

A cosa serve la inizializer list in C++? Fai un piccolo esempio e spiega il tutto tramite commenti ad un esempio di codice. Ci sono alcuni casi in cui il suo uso è facoltativo (mostra le alternative nel caso) e invece casi in cui risulta essere necessario (mostra eventualmente un esempio).