Terza lezione

15 March 2022 13:59

Dai 16 byte in poi, tutti i valori hanno segno.
$2^{16} = 65536$

Se la copia di un'elemento avviene in maniera corretta, viene copiato l'indirizzo indirizzo che ha come valore fisso 64 bit.

In allocazione:
- Per allocare un unsigned long: vado ad inizializzare il singolo valore (dandogli o meno un particolare valore)
- Il try/catch è obbligatorio almeno che non si è proprio sicuri che non si debbano lanciare eccezioni

TIPI DEFINITI DALL'UTENTE:
1) **Struct:** hanno lo stesso funzione di struct in C (stesso tipo di denominazione)
   Struct structname{
    campi
   };
- Dato che l'accesso in memoria avviene tramite i multipli di 8 byte, il primo carattere di una struct utilizzerà 8 bit ma tutto il resto verrà sprecato.
- Se si riordino i dati nella struct, a livello logico non cambia nulla ma cambia qualcosa nel footprint della memoria in quanto si stanno utilizzando al meglio quei 32 bit
- Le struct **SONO DELLE CLASSI**
- Di default sono pubbliche

2) **Enumerazione:** rappresentare valori tramite mnemonico ovvero un insieme molto piccolo di valori
   enum class Nome{elemento 1, …., elemento N};
- Se ho più di una enumerazione, non posso utilizzare due volte lo stesso elemento
- Confrontati per uguaglianza, differenza e valore

3) LIBRERIA IOSTREAM & FUNZIONI I/O
   #include <iostream>
- << da applicare a oggetti di tipo ostream
- >> da applicare a oggetti di istream

```
ostream& operator<<(ostream& os, const nomeclasse&*){
 return os<<st.name<<"-"<<st.cognome<<"-"<<st.matricola<<"-"<< endl;
}
 istream& operator>>(istream&is, nomeclasse&st){
Return is>>st.nome>>st.cognome>>st.matricola>>st.id;
```

1) Classe istream da finire
2) Classe o stream da finire

LIBRERIA STRING
- String (#include <string>) è diverso da una stringa di litterali C e C++ (char*)
- String nome= "Nome" oppure ("nome") oppure {"nome"}
- Gli oggetti string sono confrontabili lessicograficamente
- Operatori << input, >> output
- Size[]; empty []; [i]; front(); back()
- Concatenazione di stringhe è possibile farla cosi come la sottostringa

GENERAZIONE PSEUDO-CASUALE DI NUMERI
```
#include <random>
 default_random_engine gen(random_device{}());
 uniform_int_distribution <type> dist(i,j);          devo fare una chiamata ad un oggetto
 for (uint i=0; i<15, i++)
{
 cout<<dist(gen);
}
```

Devo dichiarare gli attributi privati a differenza delle classi dove lo sono a prescindere.