Prima lezione

08 March 2022 14:00

- **Professore: Fabio Mogavero**
- Testi di riferimento: quelli presenti nel file .zip
- 08 Marzo-07 giugno
- Ricevimento: lunedì (variabile), tramite Teams, 16:00-18:00

Argomenti del corso:
- Elementi del linguaggio C++
- Tipi di dati astratti e relative implementazioni
- Progettazione di una libreria contenitore di dati
- Iteratori sui dati
- Strutture dati elementari (vettori, liste, pile, code)
- Alberi binari di ricerca
- Matrici
- Grafi oppure HashMap

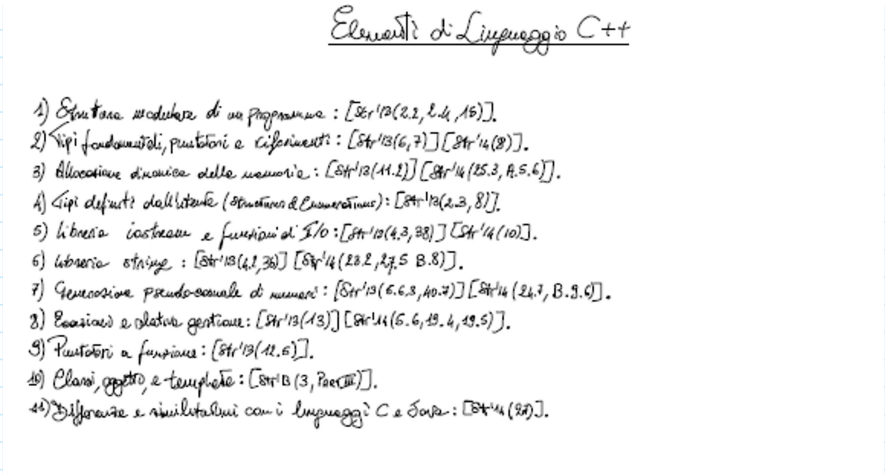
Esercizi di intercorso (con modalità di consegna sulle slides del file .zip) con file testing per testare le librerie (regole come l'anno scorso)
- 24 Marzo-10 aprile, **vettori e liste**
- 7 aprile-24 aprile, **pile e code**
- 21 aprile-8 maggio, **alberi binari di ricerca**
- 5 maggio-22 maggio, **iteratori sugli alberi**
- 19 maggio-5 giugno, **matrici**
- 2 giugno-19 giugno, **grafi**

Sulle slides: capitoli del libro dove studiare la teoria utile

Esame:
- Se si hanno consegnato tutte le librerie, verrà implementata una nuova funzione + nuova funzione (no prova a calcolatore: direttamente orale) + orale (LINGUAGGIO, domande sulle librerie)
- Esame non semplificato: scrittura da capo di una libreria vuota (no calcolatore: solo interfaccia e alcune funzionalità) da svolgere in 3 ore + orale (identico al precedente)

Imparare ad usare i file!!

La grandezza del puntatore è sempre la stessa (64 bit in architetture moderne mentre 32 bit in altre più vecchie).
- Allocazione dinamica
- Struct, ereditarietà multipla che in C++ è possibile ottenere
- Puntatore a funzione
- Oggetto, come si scrive la gerarchia di un'ereditarietà di oggetto (private, public…)
- Template



1. **Main cpp**

La prima cosa che una funzione deve avere è il main

```
#include <iostream>
 using namespace std; specificatore che ci permette di separare i nomi degli spazi(se trovo std::out devo prendere lo stream di out nello spazio di nomi della libreria)
int main (){
 cout<<"Hello world!"<< endl;
return 0; indica che è terminato in modo corretto ed è opzionale
}
```

Non può ritornare un void

Per compilare da riga di comando: g++ -O program main.cpp

2. **Struttura modulare di un programma**

Nel file main.cpp

```
 #include "test.hpp"
int main(){
 test(),     questa funzione deve essere seguita su un file diverso
 test2();
}
```

Nel file test.cpp

```
#include <iostream>
 using namespace std;
void test() {
 cout<<"Hello world!"<< endl;
}
```

DA EVITARE UN INCLUDE DI FILE CPP, solo in un caso può avvenire ovvero quando si vanno ad utilizzare i template!
Vado ad includere un header file, ovvero un file che contiene SOLO le implementazioni delle funzioni

Nell'header file test.hpp

```
 void test();
```

Da riga di comando: g++ -O program main.cpp test.cpp

I nomi dei file non sono altro mnemonici per coloro che vanno a programmare

Il metodo in cui si trova deve essere definito prima(non per forza deve essere implementato). Se ci sono più o nessuno, si ferma (nel primo caso non sa quale deve scegliere). **Per questo motivo non posso includere i file cpp.**

L'header file può essere incluso più volte senza includere direttamente l'implementazione
Posso definire un'header file con:

```
#ifdefine TEST_HPP
#define TEST_HPP
```

```
 void test()
#endif
```
Questo perché si potrebbe trovare l'inclusione di "test.hpp" e un'altra inclusione di "test.hpp" (in questo caso non va a funzionare l'header file).
Il nome di queste stringhe avranno SEMPRE nomi diversi per comodità e avranno un proprio corpo.