Esercitazione 1

UNIVERSITÀ DEGLI STUDI DI GENOVA | Dibris

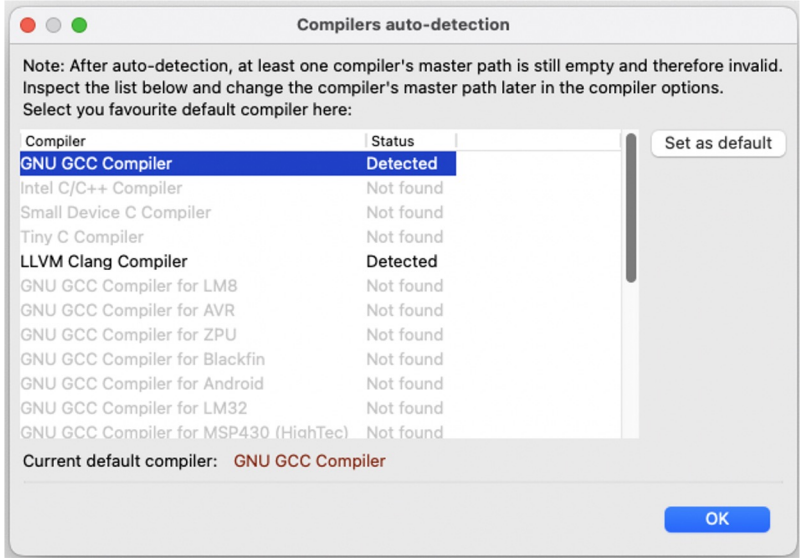
Gualtiero Volpe
gualtiero.volpe@dibris.unige.it

Esercizio 1.1

Sviluppare in linguaggio C++ un programma che chieda all'utente di immettere due numeri interi, sommi questi numeri tra di loro e stampi a video la somma.

Code::Blocks - Primo Avvio e configurazione

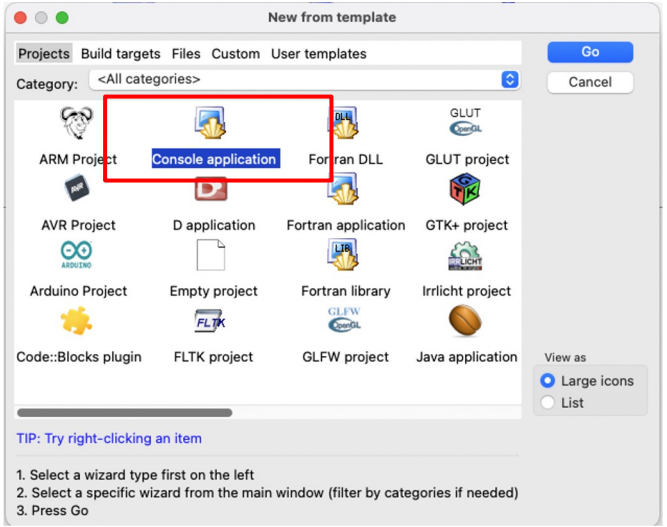
Al primo avvio verrà richiesto di scegliere un compilatore da utilizzare.
Selezionare GNU GCC Compiler, quindi su “Set as Default” e poi su “Ok”



Creazione di un progetto in Code::Blocks

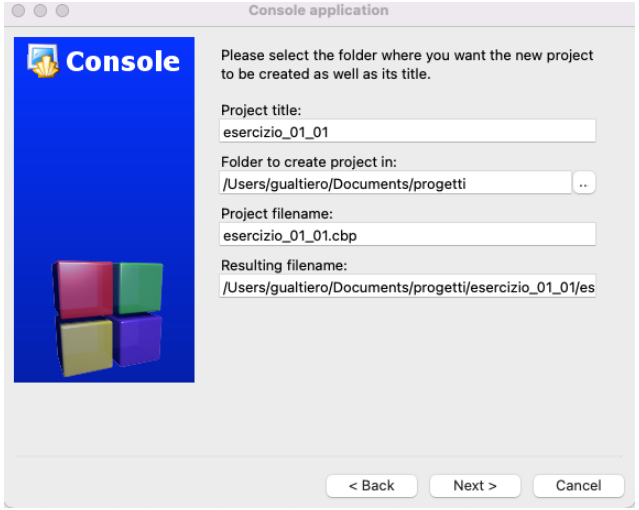
Apriamo Code::Blocks e clicchiamo su ***Create a New Project***

Nella successiva finestra, selezioniamo **Console Application** e clicchiamo **Go**



Creazione di un progetto in Code::Blocks

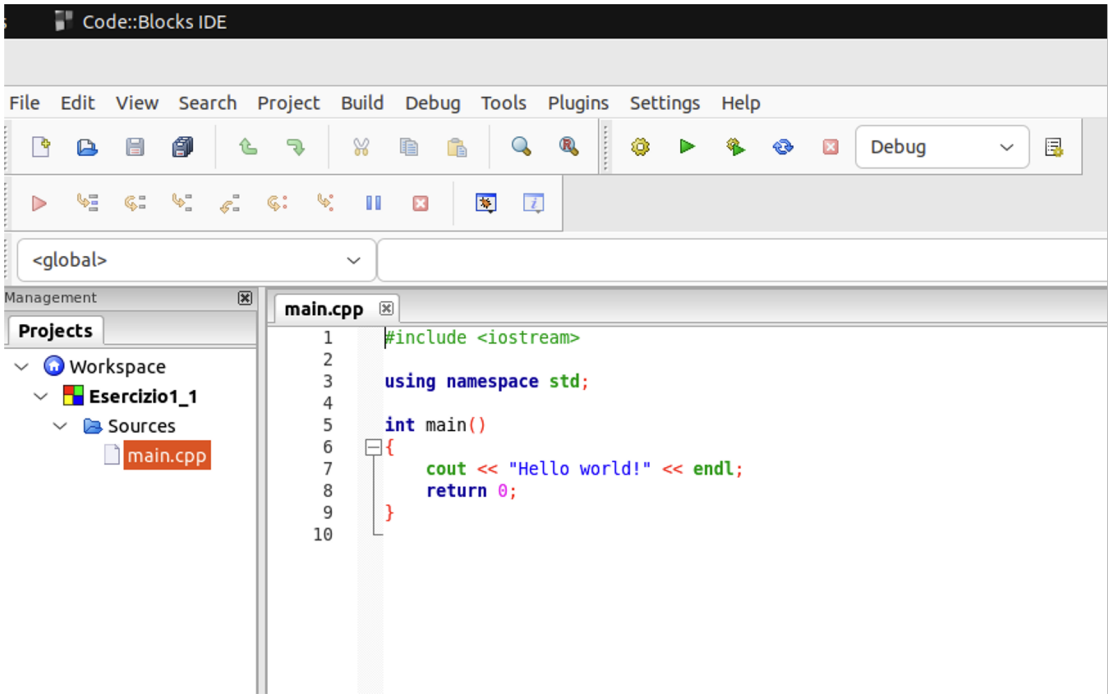
Nelle finestre successive, selezioniamo C++, e infine inseriamo il nome e il percorso del nostro progetto. Vi consiglio di creare una cartella apposita in cui inserire tutti i vostri progetti.



Continuate, selezionate le impostazioni di Default, e abbiamo così creato il nostro workspace per il Progetto esercizio_01_01.

Esercizio 1.1

Cliccando su “Sources” e poi su “main.cpp” è possibile vedere il codice sorgente predefinito generato da Code::Blocks



Esercizio 1.1

#include <iostream>

#include è una direttiva per il preprocessore, usata per includere file nel nostro programma. In questo esempio, serve ad includere il contenuto del file *iostream.*

In particolare, questo ci serve per usare le librerie standard di I/O di C++ che mettono a disposizione del programmatore quanto necessario per la gestione delle operazioni di I/O.

Questo sistema si basa sul concetto di canale (stream) inteso come mezzo attraverso cui confluiscono le informazioni provenienti o inviate dai diversi dispositivi hardware (le unità esterne).

I canali standard di I/O sono “collegati” rispettivamente agli oggetti di I/O: **cout, cerr e cin** che permettono per mezzo degli operatori **<< (inserimento)** [i cui operandi di sinistra sono rispettivamente cout oppure cerr] e **>> (estrazione)** [il cui operando di sinistra è cin] di effettuare le operazioni di I/O.

Esercizio 1.1

using namespace std

In C++ esistono i namespace ovvero costrutti che permettono di definire variabili e funzioni che siano globali, quindi visibili a tutte le funzioni, senza però entrare in conflitto con altre librerie che impiegano lo stesso nome di variabile o funzione.

Il **namespace std** definisce la variabili globali cin, cerr e cout. Se quindi voglio utilizzarle, ho due possibilità:

1. Includere il file in cui sono definiti (#include <iostream>) e usare il namespace std (using namespace std) nel codice, così da poter direttamente utilizzare quelle variabili
2. Includere il file in cui sono definiti (#include <iostream>) e fare riferimento specifico al loro namespace ogni volta che le utilizzo, usando l'operatore di risoluzione :: (es. **std::cout**)

La prima opzione **non** è considerata una best practice ma può essere utilizzata sopratuttto nelle prime esercitazioni per semplificare la scrittura del codice

Esercizio 1.1

int main() {...

return 0;

}

Ogni programma in C++ deve avere la funzione main(). Le parentesi graffe indicano l'inizio e la fine della funzione. L'esecuzione del codice inizia da questa funzione.

A norma di standard, main deve avere tipo *int* e se nel corpo della funzione non viene inserita esplicitamente un'istruzione return il compilatore inserisce automaticamente **return 0;**

Esercizio 1.1

cout << "Hello world!" << endl;

cout stampa il contenuto all'interno degli apici. Deve essere seguito dall'operatore << (inserimento), seguito dalla stringa che si vuole stampare a schermo. **endl** appartiene sempre al namespace std, serve per andare a capo, oltre che per assicurarsi che il buffer in uscita venga *scaricato.*

Due ulteriori aspetti importanti:

- In C++ la separazione tra istruzioni è indicata dal punto e virgola (;) che termina ciascuna di esse. La suddivisione del programma in più righe serve a rendere il programma più leggibile alle persone che lo leggono, per il compilatore la cosa non fa' alcuna differenza.

- Le istruzioni sono leggermente spostate sulla destra. Il codice si dice «indentato». Anche questa è solo una questione di leggibilità, ma non è importante ai fini della compilazione.

Esercizio 1.1

Procedura per risolvere l'esercizio:

1. dichiarare due variabili intere per effettuare la somma
2. utilizzare gli oggetti di I/O cin e cout per ottenere i due numeri tramite il terminale, salvando il contenuto nelle variabili corrispondenti
3. effettuare l'operazione di somma
4. stamparla sul terminale

Esercizio 1.1 - Soluzione

```cpp
#include <iostream>

int main()
{
    int number1 = 0, number2 = 0;
    std::cout << "Inserire un numero intero: ";
    std::cin >> number1;
    std::cout << "Inserire un altro numero intero: ";
    std::cin >> number2;
    int somma = number1 + number2;
    std::cout << "La somma dei due numeri vale: " << somma << std::endl;
    return 0;
}
```

Esercizio 1.1 – Soluzione alternativa

```cpp
#include <iostream>
using namespace std;

int main()
{
    int number1 = 0, number2 = 0;
    cout << "Inserire un numero intero: ";
    cin >> number1;
    cout << "Inserire un altro numero intero: ";
    cin >> number2;
    int somma = number1 + number2;
    cout << "La somma dei due numeri vale: " << somma << endl;
    return 0;
}
```

Con l’utilizzo di using namespace std

Processo di Compilazione

Per creare un programma eseguibile è necessario prima compilare il codice sorgente

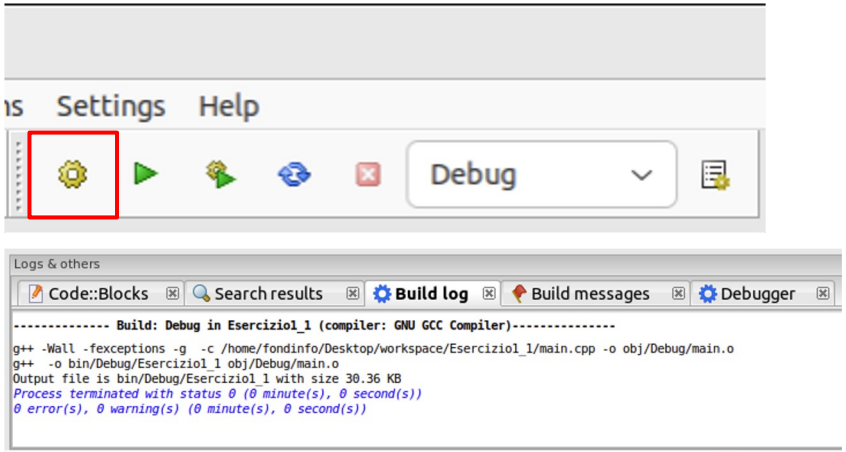
Che significa compilare?

- La compilazione traduce il codice sorgente in codice oggetto, ovvero traduce le informazioni scritte in un linguaggio di programmazione nel linguaggio macchina del computer, il codice binario.
- Questa operazione viene effettuata tramite un programma specifico, chiamato compilatore.

Compilazione del Programma

Per compilare il nostro programma, possiamo cliccare su Build (rotellina gialla, oppure nel tab Build, oppure la scorciatoia da tastiera CTRL+F9, su Windows).

Durante la compilazione, possiamo vedere l'esito della compilazione nel build log (in basso), in maniera da capire se ci sono errori, e avere suggerimenti per la loro risoluzione

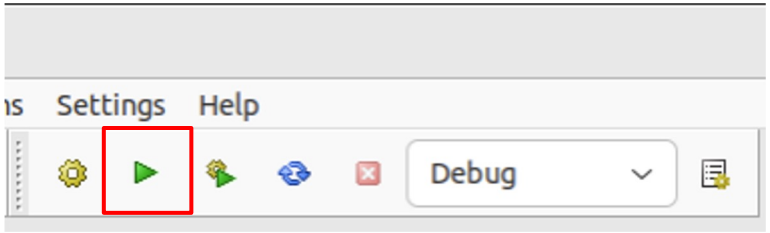


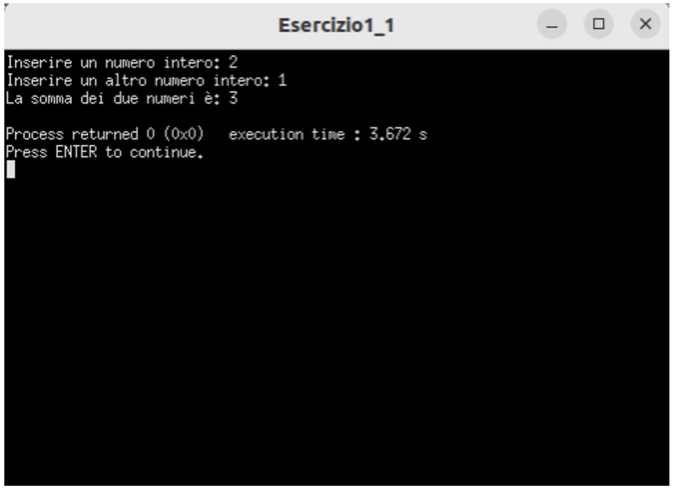
Nel caso in cui ci sia un errore con il compilatore, potete provare a risolverlo andando su:
Settings -> Compiler -> Toolchain Executables e cliccare su AutoDetect, per far sì che il compilatore, se presente sul vostro computer, venga integrato nell'ambiente di Code::Blocks

Esecuzione del Programma

Per eseguire il codice, possiamo usare il tasto Play
(o Run sotto il tab Build, o la scorciatoia da tastiera CTRL + F10).

Il terminale verrà avviato automaticamente con l'esecuzione del programma:





Esercizi da Svolgere

Esercizio 1.2: *Sviluppare in linguaggio C++ un programma che chieda all'utente di inserire cinque numeri interi e ne stampi a terminale la media (approssimandola ad un numero intero)*

Esercizio 1.3: *Sviluppare in linguaggio C++ un programma che chieda all'utente di immettere la lunghezza (intera) di due cateti di un triangolo rettangolo e stampi a terminale il quadrato dell'ipotenusa.*

Esercizio 1.4: *Sviluppare in linguaggio C++ un programma che chieda all'utente di immettere due numero interi, divida il primo numero per il secondo, e stampi a terminale il quoziente e il resto*

Esercizio 1.5: *Realizzare un convertitore Euro -> Lire, utilizzando come fattore di conversione il numero intero costante 1936.*

Istruzioni Finali

Troverete le soluzioni di questi esercizi - con qualche commento aggiuntivo - qui:

https://github.com/gualtierovolpe/fondamenti_di_informatica

Le soluzioni delle esercitazioni vengono rilasciate il giorno prima dell'esercitazione successiva.