**int main(int argc, char \*argv[])**

Il Main va trattato come una qualsiasi funzione che abbiamo visto in Fondamenti per Python. “**main**” è il nome della funzione, e tra le due parentesi abbiamo i parametri che andiamo a passare.

* **Argc** corrisponde al numero degli argomenti che andiamo a passare, nome del programma incluso. Per esempio, compilo un programma di nome “**test**”, lo mando in esecuzione con “**test.exe paramentro1**”, argc = 2, in quanto un argomento sarà rappresentato dal nome del programma, cioè test.exe, mentre un altro argomento sarà dato dal parametro passato.
* \*argv[] invece permette di accedere al valore dei parametri passati, dove argv[0] corrisponde al nome del programma. Per esempio, compilo sempre “**test**”, lo mando in esecuzione con “**test.exe parametro1**”, ho che **argv[0]** sarà uguale a test.exe, mentre **argv[1]** sarà uguale a paramentro1.

Questa struttura del main, per convenzione, si utilizza quando si lavora da Terminale, e quindi il passaggio dei parametri avviene direttamente dal terminale quando non è presente una funzione che permette l’input delle informazioni necessarie all’utente oppure quando non sono dichiarate in modo fisso all’interno della funzione o del file.

Per quanto riguarda l’int davanti ad **argc** ed il char davanti ad **\*argv[]** bisogna ricordare che, a differenza di Python dove la tipologia del parametro era opzionale e veniva inserita solamente tramite un commento, nel caso del C è obbligatorio specificarla nella stesura del codice della funzione. Cioè bisogna esattamente dire che tipo di elemento deve essere argc e che tipo di elemento deve essere \*argv[]. E ciò vale per qualsiasi tipo di funzione.

**Return 0**

La questione del “**return 0**” è un po’ più delicata. Più in generale, return 0 dice a chi effettua la chiamata del programma (il Sistema Operativo nel nostro caso) sta a significare che il programma è stato eseguito correttamente e che può procedere con il resto. Cosa sta a significare il **“procedere con il resto”**?

Nel nostro caso abbiamo solamente una funzione, per esempio quella che deve stampare a schermo semplicemente **“Hello World!”,** quindi è alquanto banale come cosa. Se però abbiamo a che fare con applicazioni molto più complesse, dove ci sono diverse chiamate di funzioni, la cosa diventa differente.

Se per esempio la funzione A chiama la funzione B, può essere necessario imporre un controllo. Per esempio:

**funzioneA:**

**[codice vario]**

**Se B ha come valore return 0, cioè B è stata eseguita correttamente, allora continua ad eseguire il resto del codice di A, altrimenti fermati ed esegui [codice di errore]**

Dove B avrà al proprio interno un codice del tipo

**funzioneB**

**se [codice vario] = []**

**allora return 0**

**altrimenti return [y] -> con y diverso da 0 (Bisogna ricordare che qualsiasi valore diverso da 0, viene interpretato come una non riuscita dell’esecuzione del programma)**

In ogni caso, per prendere la mano, è sempre consigliato inserire **return 0** in tutti gli esercizi o codici che si fanno al PC per prendere la mano, in quanto è buona prassi non ometterlo mai nel codice. Ed è anche il motivo per il quale scriviamo **int main**, in quanto la funzione **main** solitamente restituisce un valore numerico (0 nel nostro caso) ed alcuni compilatori segnalano come errore la sua omissione. (Gcc compila tranquillamente, però emette un Warning)

**Void ed int main(void)**

Per quanto riguarda **main(void),** semplicemente dichiariamo che la funzione **main** non dipende da nessuna variabile “esterna”. Se per esempio ho la funzione che mi stampa “Hello world!” vista il primo giorno a lezione, non ho dipendenza da nessun parametro e quindi posso tranquillamente scrivere **main(void).** Un contro esempio può essere ciò che abbiamo visto sopra. Se io passo delle variabili alla funzione tramite terminale, per esempio, non posso utilizzare **main(void)** in quanto nel codice visto nella seconda lezione, per esempio, io eseguo il controllo sulla lunghezza, il for e tutto il resto sui parametri passati tramite terminali, e devo dunque dichiararli nella stesura del codice della funzione. (Per farla breve, in Python era nomefunzione(), cioè senza specificare nessun paramentro)

**void nomefunzione()** invece viene utilizzato nel caso in cui la funzione non deve restituire nessun valore. Per esempio una funzione che deve stampare un numero, una stringa o che magari deve disegnare una figura geometrica, non deve restituire nessun valore e quindi posso utilizzare la parola chiave **void** davanti.