Masterclass di (NON) Programmazione C/C++ Embedded & Realtime

Sviluppo Software di applicazioni ***Business & Safety-Critical,*** *per* ***Sistemi Embedded, Real-Time, IoT (Internet of Things), App mobili***

WWW.SOFTWARESICURO.IT

Me l'avete chiesto veramente in tanti, tantissimi: ho capito che c'era un tassello fondamentale che mancava nella mia offerta di consulenza e corsi.

Infatti, man mano che parlavo con voi ho capito che c'era tutta una serie di concetti che io davo per scontati, avendoli appresi in diversi contesti e applicazioni internazionali in oltre 30 anni di carriera, che quindi non potevano essere necessariamente noti a tutti.

Per cui, eccolo qua: il mio ***Corso*** o meglio ***una MasterClass di Programmazione Embedded & Realtime,*** *per applicazioni* ***Business- e Safety-Critical,*** anzi il primo corso al mondo di ***NON Programmazione!***.

E se segui, capirai cosa vuol dire e perché è diverso da tutti i corsi che hai fatto finora, per cui non è il caso assolutamente di perdertelo: infatti, tieni conto che è ***il distillato di tutta la mia esperienza****,* è *il corso che avrei voluto avere a disposizione quando ho cominciato a lavorare ma non c'era (e pure adesso, direi che è il primo nel suo genere).*

Si torna alle basi: questo corso farà sicuramente parte di quello che io chiamo gli ESSENTIALS, ossia tutti quei corsi, infoprodotti, programmi di apprendimento dei concetti di base che ho deciso di creare, per dare la possibilità anche a te di fare molto più velocemente il percorso che ho fatto io, senza ripetere gli stessi errori.

Esatto, questo è il punto chiave: ***evitare tutti gli errori che ti portano ad avere un Software NON Sicuro.***

Per chi è questo corso?

Un corso quindi ***intensivo*** ed essenzialmente per ***PROGRAMMATORI E ARCHITETTI SOFTWARE*** *o aspiranti tali:* può essere tranquillamente ***il tuo primo corso di programmazione***, senza saperne nulla prima.

Magari fosse stato il mio primo corso! Quante notti insonni, quante incazzature, quante problematiche avrei evitato...

Ma contiene così tanti concetti avanzati e poco insegnati, frutto della mia esperienza personale che oramai supera e di molto i 30 anni (sì, ho iniziato a programmare il BASIC e il PASCAL tipo a… 12-13 anni!) in centinaia di progetti e di clienti in tutto il mondo, dal gestionale all'aerospaziale, che tutto quel che ho appreso e concentrato in questo ***può essere di grandissima utilità***

anche a chi ha magari un sacco di anni di esperienza.... perché ti potrebbe evitare le testate contro il muro prese da me e dai miei colleghi e partner in questi anni in ambienti estremamente severi e critici.

E' anche un corso per **MANAGER:** al di là degli aspetti tecnici di dettaglio, è un compendio di *best practises, consigli strategici, impostazioni iniziali corrette ed errori da evitare.* Come capirai continuando a leggere, grazie al distillato di esperienza personale nelle aziende e progetti in cui ho lavorato da dipendente, consulente e formatore, non è solamente un classico corso di programmazione, è una vera e propria ***guida strategica alla progettazione e scrittura di Software Sicuro.*** Per cui è fondamentale che siano concetti che padroneggi anche tu, per poterli implementare con il tuo team.

Perché un altro corso?

Ma se già programmi da tempo, perché mai dovresti rifare un corso di programmazione dopo anni dall'ultima volta che ne hai fatto uno?

Semplice: perché questo è il corso che avrei voluto fare io all'inizio della mia carriera ma non esisteva (e non esisteva fino ad ora!)*, invece di quello che "passava il convento":*

- *manuali strettamente tecnici e didascalici*
- *corsi strutturati come un mero elenco di costrutti, strutture e dichiarazioni...*
- *prove su prove su prove, errori inspiegabili, crash disastrosi*
- *rifacimento di intere applicazioni e architetture per colli di bottiglia e problemi strutturali*
- *notti insonni, straordinari al lavoro, ritardi, problemi col cliente...*

Oltre ad essere un corso tecnico, molto intenso e fitto di informazioni, è fondamentalmente un corso strategico: ***ti insegna il mindset corretto, l'impostazione iniziale, le scelte chiave, l'approccio necessario ad evitare grossi guai futuri.***

Non sto a ripetere la frase del grandissimo Michael Jordan, di cui è uscito di recente il film a puntate "The Last Dance" su fatto che lui stesso, per diventare il più grande, ha imparato dai propri errori.

Quello che otterrai in questo corso senza respiro è proprio *l'elenco delle cazzate meravigliose che ho commesso io (o visto commettere), delle scelte in apparenza intelligenti e poi rivelatesi disastrose, di tutti i "****mai più!****" che ho dovuto esclamare nella mia carriera e di conseguenza tutto quello che avrei dovuto fare* ***(ed è il caso che cominci almeno a fare TU).***

SOFTWARE SICURO

Cos'ha di diverso questo corso?

La differenza di questo rispetto ad altri N-mila corso esistenti al mondo, è proprio il fatto che è ***PREVENTIVO***: non ti spiega all'inizio una serie di tecniche in fila, tra cui alcune estremamente pericolose e inefficienti, per poi scoprire solo dopo mesi o anni (se va bene in un corso avanzato, più spesso sulle tue spalle le conseguenze disastrose di certe scelte).

Ci sono cose nei linguaggi storici e moderni che sono estremamente rischiose e vanno usate con estrema cautela: perché allora insegnarle? Perché imparare a fare la nitroglicerina in casa, quando rischi di saltare in aria te, la tua casa, la tua famiglia e tutto il vicinato?

Fin da primo giorno, ***scoprirai non solo i dettagli tecnici,*** ma anche e soprattutto:

- *Le **scelte strategiche** che devi fare prima di scrivere la prima linea di codice (**linguaggio**, **sistema operativo**, **microprocessore**, **architettura**, …)*
- *I compromessi e le conseguenze di certe **decisioni di base** che si ripercuoteranno in futuro*
- *Quali **tecniche**, **costrutti**, **tecnicismi** sono sicuri e affidabili e quali invece richiedono cautela e pianificazione, o sono da evitare del tutto*
- *Quali sono le **Best-Practises**, le **linee guida**, gli **standard internazionali** per lo sviluppo di Software più sicuri e affidabile*
- *Come evitare le sirene del marketing di **tool** e **altre diavolerie** che ti fanno andare fuori strada promettendo risultati irraggiungibili*
- *Gli **errori più comuni**, i **trabocchetti**, gli **effetti collaterali** da evitare a tutti i costi*

Senza tutte queste informazioni fondamentali a completamento delle nozioni tecniche, ***sarebbe come insegnarti a cosa serve ogni singolo bottone, leva e manovella di un jet di linea, senza avere comunque la minima idea di cosa voglia dire pilotare un aereo.***

Che applicazioni potrai sviluppare dopo questo corso?

Che tipo di applicazioni potrai sviluppare dopo un corso del genere?
Serve per fare app per telefonino o un software aeronautico questione di vita o di morte?
Meglio per programmare un gestionale oppure un delicatissimo dispositivo medicale?

Questo è un modulo formativo **ESSENTIALS**: quindi pur essendo ad altissima densità di informazioni rimane comunque un corso base, non è dedicato a un dominio specifico, un linguaggio preciso, un'architettura particolare. Non fa moltissima differenza se sviluppi ***app per telefonini, dispositivi embedded, sistemi gestionali***.

*I principi di base dello Sviluppo di Software Sicuro sono **ESATTAMENTE GLI STESSI** e vanno solo calati nella realtà tecnica in cui operi.*

Per questo motivo, si tratta di un corso che ti potrebbe rendere già autonomo ed essere in grado di programmare quello che vuoi, ma richiede poi degli approfondimenti specifici, più tecnici e di dettaglio, che sarò lieto di suggerirti.

Detto questo, posso consigliarti questo corso se ricadi in uno di questi casi:

- *Se sviluppi applicazioni che devono avere **un alto grado di affidabilità e sicurezza** (es. che trattano **dati sensibili**, **transazioni economiche**, oppure con **tantissimi utenti**)*
- *Se lavori in un settore **Safety-Critical** dove la sicurezza non è un optional (**aerospazio, medicale, automotive, …**)*
- *Se vuoi in generale **incrementare l'Affidabilità e l'Efficienza** del tuo processo di Sviluppo Software*
- *Infine, se lavori in maniera totalmente dedicata al Cliente per cui vuoi fornire un Software che **massimizzi la Customer Satisfaction***

Con questo bagaglio culturale immenso, condensato fitto fitto in una manciata di moduli di formazione, ***ti renderà già un programmatore evoluto e con un livello di esperienza e di padronanza della programmazione software che nessun altro corso "teorico" potrebbe mai darti.***

SOFTWARE SICURO

1° Corso di Programmazione Business- e Safety-Critical

Il 1° corso di NON programmazione*: non solo le basi della scrittura del codice, ma soprattutto le Best-Practises, gli errori più comuni, la scelta della piattaforma, la gestione del real-time e dell'embedded e tutti gli altri aspetti più critici.*

Sviluppo Software per Sistemi Embedded, Real-Time, IoT (Internet of Things), App mobili

Struttura: 12 moduli online, da 3 ore circa ciascuno, bi-settimanali, interattivi tramite webinar ad accesso individuale (webcam richiesta).

Partenza: Agosto-Settembre 2020 (ci si può iscrivere anche successivamente).

Prezzo e altre informazioni:

- *Mail: sales@softwaresicuro.it*
- *Telefono: 349 673 6781 (Anna Chiara Cesari – Marketing & Sales)*

Materiale compreso:

- *Lezioni online*
- *Slides cartacee*
- *Altro eventuale materiale didattico*
- *Community online riservata agli studenti per domande ai docenti, scambi di pareri, ecc.*
- *Esame finale*
- *Certificato di partecipazione e superamento esame in pergamena*

Materiale opzionale: registrazioni video delle lezioni in cofanetto di lusso

SOFTWARE SICURO

ARGOMENTI

Scelta del Linguaggio: C o C++? Java o PHP? E perché non Ada? La scelta del linguaggio è un passo fondamentale, che si ripercuote pesantemente su tutte le scelte successive, determinando limitazioni o opportunità. Una panoramica dei pro e dei contro di ogni linguaggio a seconda del contesto (embedded, applicativo, gestionale) e qualche suggerimento pratico.

Scelta del Sistema Operativo: altra scelta strategica iniziale… una volta i sistemi operativi erano pochi e la scelta era facile, oggi invece qualunque dispositivo esistente, dalla lavatrice all'orologio, ha al suo interno un sistema operativo. Panoramica delle famiglie esistenti di sistemi operativi e qualche strategia collaudata per decidere quale usare (o di NON usarlo del tutto!).

Scelta del Microprocessore: un altro momento decisionale critico, legato ai precedenti contesti. Oramai le famiglie di microprocessori sono letteralmente infinite, una vera e propria giungla di sigle, architetture, caratteristiche, configurazioni. E pure in questo caso, una vera e propria guida per districarsi e fare delle scelte consapevoli e durature, con l'ottica della compatibilità, consumi, ingombri (specialmente nel campo IoT)

Processo: quale approccio scegliere al ciclo di vita? Il vecchio Waterfall è veramente morto? Tutti usano il V-Cycle, perché? Tutti vorrebbero usare AGILE ma chi lo usa veramente e come si trova? Molto importante gettare le basi giuste e conoscere pregi e difetti dei vari modelli.

Requisiti: ne ho parlato una marea di volte, ho creato pure il corso Requisito Perfetto… ma stavolta si riparte dalla base. Che cos'è un Requisito? A cosa serve? Come si scrive? E da lì si innesta buona parte del contenuto del mini-corso Requisito Perfetto

La scelta dell'Architettura: non è mica finita con le scelte strategiche di cui abbiamo parlato finora. Abbiamo sistema operativo, linguaggio, microprocessore… ma che architettura interna ed esterna adotteremo? Quali pattern? Che design? Architettura distribuita, master-slave, IoT, cloud… ce ne sono tantissime ed è importante capire da subito le conseguenze di ogni scelta.

Stile di programmazione: sì, anche se non stiamo scrivendo un romanzo… è una questione di stile anche qua. Linee-guida, costrutti permessi e vietati, standard come MISRA, JSF, JPL e altre Best-Practises vengono affrontati perché importanti per quello che si scriverà dopo.

Dichiarazioni delle variabili: andiamo sul concreto della programmazione vera e propria. Variabili, tipi, template… a cosa servono, come si dichiarano, come si usano e in che modo certe decisioni prese possono aiutarci o ostacolarci successivamente.

Allocazione della memoria: un tema immenso, che richiederebbe un corso dedicato solo ad esso. La memoria ora è abbondante quasi ovunque, ma rimangono dispositivi minuscoli molto limitati… saper padroneggiare la sua allocazione è importante, perché non sono i Giga e Gigabyte di RAM a salvarci, ma saper padroneggiare il suo utilizzo e ottimizzarlo.

Controllo del flusso: cosa offrono i vari linguaggi e sistemi operativi come costrutti di controllo del flusso? Dal classico IF-THEN-ELSE, ai vari tipi di LOOP, fino a quello che di più recente offrono i linguaggi, chiarendo subito cosa conviene (o NON conviene) usare.

Real-time: un'analisi dei meccanismi di CONCORRENZA e MULTI-TASKING tra cui MONITOR, SEMAFORI, ecc. e relative problematiche di RACE CONDITION, DEADLOCK e altre problematiche del real-time. Un modulo assolutamente fondamentale per rendere stabile e affidabile tutto il resto della nostra applicazione o dispositivo.

Comunicazione tra moduli: indipendentemente dal concetto di modulo (che sia un task, un file, un processo o un dispositivo intero), esso inevitabilmente dovrà comunicare con altri moduli, locali o remoti. E le tecniche di comunicazione devono andare di pari passo con il resto della tecnologia, come prestazioni e affidabilità.

Defensive Programming e prevenzione dei problemi: i guai, quando si comincia a programmare, si sa che arrivano a raffica. Meglio proteggersi e iniziare con il piede giusto: le tecniche, le strategie, le linee guida, le best-practises per proteggersi dai problemi di vario tipo e assicurarsi di esserne il più possibile immuni, prevendendo con la programmazione difensiva i guai più comuni.

Tools: tutto il ciclo di vita della produzione di codice è oramai accompagnato da una serie di tool. Si tratta di una lama a doppio taglio: se gestiti correttamente e introdotti nella giusta fase del

processo, con la giusta formazione agli utilizzatori, possono accelerare di parecchio lo sviluppo, altrimenti rischiano di diventare dei colli di bottiglia o peggio abbandonati nel dimenticatoio. I principali tool da utilizzare nelle varie fasi e perché.

Security: aprendo i dispositivi al contatto con il mondo esterno, come sta succedendo sempre di più nel mondo IoT, si creano una innumerevole serie di problemi di controllo degli accessi e prevenzione degli abusi. Fondamentale impostare fin dall'inizio un'architettura, un approccio, una protezione che sia by design, oltre che attiva.

Testing: il mio grande cavallo di battaglia, se mi conosci sai che potrei parlarne per settimane. Senza di questo tutto il resto non esiste, quindi cercherò in questo modulo di passarti tutto quello che ti serve per verificare di aver fatto tutto il possibile per assicurarti che il tuo software faccia (solo) quello che dovrebbe veramente fare. Solo questo modulo, vale il prezzo del corso.

Modellazione: oramai una buona parte del codice non è scritto a mano, ma generato automaticamente. Ma non solo: la modellazione risolve un problema assolutamente fondamentale, quella della scrittura dei requisiti formali. Per non parlare poi dell'aspetto più interessante: la simulazione, ossia la possibilità di testare i propri requisiti, prima ancora di scrivere il codice.

Certificazione: non sempre è obbligatoria, ma se sviluppi per ambienti Safety-Critical (Avionica, Automotive, Ferroviario, Medicale, ecc.) sei costretto a sottostare ad una serie di regole e di processi che dal tuo punto di vista ti complicano la vita, ma salvano (la vita) delle persone. Bene, sfatiamo un po' questo mito e cerchiamo di capire come certe best-practises della certificazione, siano utili anche per altri ambiti non strettamente critici.