# Test di valutazione – Modulo 1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Beatrice |
|  |  | Cognome | Testa |
|  |  | Data | 11/10/2019 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.  
ATTENZIONE: Le domande a risposta multipla possono contenere più risposte corrette.

1. *Quali tra i seguenti è un IDE? Per quelli che non sono IDE, spiegare per quale motivo non è ritenuto tale*

Notepad

Notepad++ non è un IDE, è solo un editor di testo, in cui è possibile editare codici sorgenti. Un IDE in genere incorpora diversi dei seguenti elementi in un ambiente GUI: editor del codice sorgente, compilatore, linker,

debugger, profiler, ecc.

Visual Studio Code

Anche VS Code non è un IDE, ma è un [editor](https://it.wikipedia.org/wiki/Editor_di_testo) di [codice sorgente](https://it.wikipedia.org/wiki/Codice_sorgente) sviluppat da [Microsoft](https://it.wikipedia.org/wiki/Microsoft) per [Windows](https://it.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://it.wikipedia.org/wiki/Linux) e [macOS](https://it.wikipedia.org/wiki/MacOS).

.NET

.NET è una piattaforma di sviluppo con finalità generali che offre varie funzionalità chiave, ad esempio il supporto per più linguaggi di programmazione, modelli di programmazione e l'interoperabilità nativa, che consente l'esecuzione di un'ampia gamma di scenari su più piattaforme.

Visual Studio

VS è considerate l’IDE più popolare and usato per sviluppare software. Esso supporta C, C++, C++/CLI, C# e molti altri

linguaggi di programmazione.

1. Cos’è una funzione?

Un blocco di codice che esegue determinate istruzioni

È un’eccezione non gestita

È una lista di elementi

È una struct con determinati livelli di accesso

1. *Cosa sono le proprietà, le costanti e le variabili? Dare una descrizione dettagliata di ciascun elemento.*

Proprietà: Le proprietà permettono il controllo degli accessi alle variabili di una determinata classe, favorendo l'accesso dall'esterno. Le proprietà sono assimilabili a delle combinazioni tra variabili e metodi. Nonostante non permettano il passaggio di parametri, danno comunque la possibilità di processare il valore prima che venga effettivamente assegnato ad una variabile. Una proprietà consiste di due parti, rappresentate dai metodi get e set, raccolte al suo interno.

Costante: sono delle variabili che non possono essere modificate o ridefinito.

Variabili: In C# una variabile viene dichiarata in questo modo: <data type> <name>. Le variabili permettono ti immagazzinare dati di vario tipo, ad esempio stringhe di testo, numeri od oggetti. Non esistono variabili globali, ma ci sono variabili locali che sono accessibili all'interno del metodo dove sono state definite, poi ci sono i campi della classe ai quali si può accedere da tutti i metodi della classe e anche al di fuori di essa, se la visibilità lo consente. Possono essere bool, int, float, double, string, char, etc...

1. *Cosa sono gli operatori logici e a cosa servono? Fornire una serie di esempi di utilizzo in codice C#.*

Op. di uguaglianza ==: il confronto di due valori può ovviamente essere fatto in molti modi, ma per verificare se sono effettivamente uguali, è possibile utilizzare l'operatore double-equal-sign (==)

int val1 = 4;

int val2 = 4;

if (val1 == val2)

Console.WriteLine (val1 + "è uguale a" + val2);

Op. di disuguaglianza !=: val1 != val2, val1 non è uguale a val2

Op. di onfronto (<, =< e >, =>): sfruttano i simboli di maggiore e minore per stabile se un valore è più grande o più piccolo di un altro valore.

Op logici booleani: Operatori binari & (AND logico) e | (OR logico). Questi operatori valutano sempre entrambi gli operandi. Operatori binari && (AND condizionale logico) e || (OR condizionale logico). Questi operatori valutano l'operando di destra solo se è necessario.

(val1=1 && val1=2): val1 può valere 1 e 2; (val1=1 || val1=2): val1 può valere 1 o 2

1. *Elencare quante più differenze possibili da .NET Core e .NET Framework?*

Per la creazione di applicazioni lato server con .NET sono supportate due implementazioni: .NET Framework e .NET Core. Entrambe condividono gran parte degli stessi componenti e possono condividere codice. Esistono, però, differenze fondamentali tra le due opzioni e la scelta dipende dall'obiettivo che si vuole conseguire.

.NET Framework: distribuito dal 2000, non è multi-piattaforma, supporta solo applicazioni Windows, usa Base Class library. Si utilizza generalmente per applicazioni già sviluppate, dà garanzie di stabilità ma poca innovazione.

.NET Code: è modern (2016), è una piattaforma di sviluppo cross-platform (sviluppata per Windows, macOS e Linux), è flexible deployment (side by side o machine wide), è open source su github supportato da Windows), eredita Base Class library da .NET Framework.

1. *Qual è la differenza tra una proprietà, un metodo o un campo “private” e “public” in una classe?*

*Private* e *public* definisco dei livelli di accessibilità e di visibilità in una classe.

L'accesso privato in una classe è il livello di accesso più restrittivo. I membri privati sono accessibili solo all'interno del corpo della classe o dello struct in cui sono stati dichiarati. Fare riferimento a un membro privato all'esterno della classe o dello struct in cui è stato dichiarato genera un errore in fase di compilazione.

L'accesso pubblico è il livello di accesso più permissivo. Non esistono restrizioni per i membri dell'accesso pubblico.

1. *Quali sono le differenze tra Do-While, While, For, For-Each e Switch? Fornire una descrizione ed un esempio di ciascuno di essi*

L'istruzione while esegue un'istruzione o un blocco di istruzioni mentre un'espressione booleana specificata restituisce

true. Poiché tale espressione viene valutata prima di ogni esecuzione del ciclo, un ciclo while viene eseguito zero o più

volte. Questo comportamento è diverso dal ciclo do-while, che viene eseguito una o più volte. In qualsiasi punto

all'interno del blocco while è possibile uscire dal ciclo usando l'istruzione break

.L’istruzione for valuta una sequenza di espressioni di inizializzazione e quindi, mentre una condizione è true, esegue

ripetutamente un'istruzione incorporata e valuta una sequenza di espressioni di iterazione.

L' istruzione foreach enumera gli elementi di una raccolta, eseguendo un'istruzione incorporata per ogni elemento

della raccolta.

L'istruzione switch seleziona per l'esecuzione di un elenco di istruzioni con un'etichetta switch associata che

corrisponde al valore dell'espressione switch.

Si riportano alcuni esempi di iterazioni utilizzate nei codice:

Console.WriteLine("Iterazione rubrica (for):");

for (int i = 0; i < rubrica.Length; i++)

{

Console.WriteLine(rubrica[i].FirstName + ", " + rubrica[i].LastName);

}

Console.WriteLine("Iterazione rubrica (while):");

int index = 0;

while (index < rubrica.Length)

{

Console.WriteLine(rubrica[index].FirstName + ", " + rubrica[index].LastName);

index = index + 1;

}

Console.WriteLine("Iterazione rubrica (foreach):");

foreach (Person current in rubrica)

{

Console.WriteLine(current.FirstName + ", " + current.LastName);

}

//Selezione della funzione da avviare

switch (selezione)

{

case 1:

Console.WriteLine(“case 1”);

break;

case 2:

Console.WriteLine(“case 2”);

break;

default:

Console.WriteLine("Selezione non valida");

break;

}

1. *Cosa sono le eccezioni, a cosa servono e quando non dovrebbero essere mai utilizzate?*

Le eccezioni forniscono un modo strutturato, uniforme e indipendente dai tipi di gestire a livello di sistema sia a

livello di applicazione condizioni di errore. Il System.Exception classe è il tipo di base di tutte le eccezioni. Le eccezioni

vengono gestite dall’istruzione try. L’eccezione dipende dal sistema, non dall’utente. Se è l’utente ad inserire un

numero quando viene chiesta una stringa, questa non è da considerare eccenzione perchè può essere gestita nel

codice.

1. *Qual è la differenza tra un Array e una Lista? Fornire un esempio di ciascuno di essi, come si esegue la creazione e la popolazione di queste strutture.*

Gli array funzionano come raccolte di elementi, ad esempio stringhe. Si possono usare per raccogliere elementi in un

singolo gruppo ed eseguire varie operazioni su di essi, ad esempio l'ordinamento. Gli array sono sequenze ad accesso diretto. Gli array sono dichiarati in modo molto simile alle variabili, con un set di parentesi [ ] dopo il tipo di dati, in questo modo:

int [] numbers = new int [5] {4, 3, 8, 0, 5}

In questo modo abbiamo creato un array con una dimensione di 5 di oggetti di tipo int. In questo caso la dimensione

dell’array è 5; quando si dichiara un array lo si deve dimensare, a differenza della lista.

La List può fare molte delle stesse cose di un array, ma in molte situazioni, la lista è più semplice e facile da gestire, a

discapito però della memoria. Ad esempio, non è necessario creare un elenco con una dimensione specifica, ma è

sufficiente crearlo e .NET lo espanderà automaticamente per adattarsi alla quantità di elementi man mano che se ne

aggiungono. Le liste sono contraddistinte dalle parentesi <>. Le liste sono sequenze ad accesso a partire dall’inizio o

dalla fine. Ad esempio:

List <string> dati = new List<string>();

String.Add ("una stringa"); //in questo modo si aggiumge una stringa

1. *Esercitazione:*

*Si chiede di creare un app console in .NET Framework Core 3.0 che in fase di esecuzione richieda l’inserimento di un numero compreso da 1 e 10; dopo la lettura del numero, dovranno essere creati un pari numero di oggetti “prodotto” (classe Product), ciascuno dei quali caratterizzato da codice (alfanumerico, campo “Code”) e nome (campo “Name”). Una volta terminato l’inserimento dei prodotti, gli stessi devono essere stampati a video e scritti all’interno di un file di testo in modo tale da poter essere facilmente riletti (la funzione di rilettura non è richiesta).*

[Tot: \_\_/10]