Gestione degli archivi e stampe

Utilizzando il linguaggio Visual Basic, con la gestione tradizionale degli archivi, sono consentiti tre tipi di accesso ai file:
- **sequenziale**, per la lettura e scrittura di file di testo
- **casuale** (*Random*), per la lettura e scrittura di file strutturati come record di lunghezza fissa
- **binario** (*Binary*), per la lettura e scrittura di file strutturati in modo arbitrario.

L'accesso **binario** consente di utilizzare file per memorizzare dati di qualsiasi tipo: con questa modalità si ha il massimo controllo su un file, in quanto i byte del file possono rappresentare qualsiasi cosa. Inoltre l'accesso binario offre il vantaggio di costruire file aventi dimensioni ridotte, risparmiando spazio su disco, quando si utilizzano record di lunghezza variabile.
L'accesso è simile a quello casuale: tuttavia, poiché le registrazioni sono considerate una sequenza di bit e i campi sono di lunghezza variabile, in fase di lettura non è possibile riferirsi direttamente ad un record e quindi i record possono essere ritrovati solo in modo sequenziale.

Un file può essere utilizzato per scrivere nuove registrazioni, per leggere dati o per aggiungere dati in coda a quelli già registrati.
La scrittura a partire dall'inizio, eseguita su un file già esistente, provoca la cancellazione di eventuali dati presenti nell'archivio.
Nei programmi di lettura in un archivio sequenziale, occorre introdurre un opportuno controllo per segnalare il raggiungimento della fine del file.

Prima di eseguire sul file una qualsiasi operazione, è necessario aprirlo con la funzione **FileOpen**: essa assegna un **buffer**, cioè un'area di memoria centrale per eseguire le operazioni di I/O sul file, stabilisce la modalità di accesso e imposta eventuali altri parametri.

La sintassi completa della funzione **FileOpen** per l'apertura di un file è la seguente:

FileOpen (*NumeroFile*, *NomePercorso*, *ModalitàAccesso*, *Permessi*, *Condivisione*, *LunghezzaRecord*)

- *NumeroFile* è un numero compreso tra 1 e 511 inclusi, che identifica il file all'interno del programma; questo è importante perché in uno stesso programma possono essere aperti e utilizzati più archivi.
- *NomePercorso*, o *pathname*, specifica il nome del file e può includere la cartella (*directory*) e l'unità a dischi.
- *ModalitàAccesso* specifica la modalità di accesso al file: *Append*, *Binary*, *Input*, *Output* o *Random*.
- *Permessi* specifica le operazioni consentite sul file aperto: *Read*, *Write* o *ReadWrite*. Il permesso predefinito (valore di *default*) è *ReadWrite*.
- *Condivisione* specifica le modalità applicate al file (condiviso, *Shared*, o bloccato, *Lock*) quando è utilizzato da utenti diversi: *Shared*, *Lock Read*, *Lock Write* e *Lock ReadWrite*. La modalità predefinita (valore di *default*) è *Lock ReadWrite*.
- *LunghezzaRecord* specifica il numero di byte del record, è un numero minore o uguale a 32.767.

I primi tre parametri sono obbligatori, gli ultimi tre sono facoltativi.
Per conoscere il numero di file disponibile si usa la funzione **FreeFile** che restituisce un valore di tipo *Integer* utilizzabile come primo parametro dalla funzione *FileOpen*.

```
Dim NumeroFile As Integer
NumeroFile = FreeFile()
```

La modalità di accesso (terzo parametro) è specificata attraverso le costanti dell'insieme **OpenMode** (*enumerazione* di Visual Basic) secondo i valori della seguente tabella:

Costante	Descrizione
OpenMode.Append	File aperto per l'accodamento
OpenMode.Binary	File aperto per l'accesso binario
OpenMode.Input	File aperto per l'accesso in lettura
OpenMode.Output	File aperto per l'accesso in scrittura
OpenMode.Random	File aperto per l'accesso casuale

Quando si apre un file in modalità *Input* o *Append*, è necessario che il file esista, altrimenti viene generato un errore. Quando si usa la funzione *FileOpen* e modalità *Output* con un file che non esiste, il file viene prima creato e quindi aperto.

L'accesso più semplice ai file è di tipo **sequenziale**, nel quale i dati vengono scritti e letti uno di seguito all'altro.
L'accesso **binario** (*Binary*) è utilizzato quando è importante che le dimensioni del file rimangano limitate. Questo tipo di accesso non richiede campi di lunghezza fissa, quindi si creano record di lunghezza variabile che consentono di conservare spazio su disco.
L'accesso **casuale** (*Random*) è sinonimo di accesso diretto, cioè si può accedere al record specificandone la posizione all'interno del file. Per questo motivo questi tipi di file hanno una lunghezza fissa di record, che viene specificata come ultimo parametro nella funzione *FileOpen*. Nell'accesso **sequenziale** la lunghezza del record è uguale al numero di caratteri memorizzati nel buffer di I/O.

I permessi (quarto parametro) sono specificati attraverso l'enumerazione **OpenAccess** secondo i valori della seguente tabella:

Costante	Descrizione
OpenAccess.Default	Lettura e scrittura consentite (impostazione predefinita)
OpenAccess.Read	Lettura consentita
OpenAccess.ReadWrite	Lettura e scrittura consentite
OpenAccess.Write	Scrittura consentita

La condivisione (quinto parametro) è specificata attraverso l'enumerazione **OpenShare** secondo i valori della seguente tabella:

Costante	Descrizione
OpenShare.Default	LockReadWrite (impostazione predefinita)
OpenShare.LockRead	Il file non può essere letto da altri processi
OpenShare.LockReadWrite	Il file non può essere letto o scritto da altri processi
OpenShare.LockWrite	Il file non può essere scritto da altri processi
OpenShare.Shared	Il file può essere letto o scritto da qualsiasi processo

Dopo aver aperto un file in modalità *Input*, *Output* o *Append*, è necessario chiuderlo con la funzione **FileClose** prima di riaprirlo per l'esecuzione di un altro tipo di modalità.
La sintassi della funzione *FileClose* è

FileClose(NumeroFile)

NumeroFile identifica il file da chiudere, attraverso il numero assegnato ad esso con la funzione *FileOpen*. Se si omette il valore del numero di file vengono chiusi tutti i file attualmente aperti. La funzione *FileClose* interrompe la corrispondenza tra un file e il suo buffer in memoria, liberando lo spazio per esso riservato: se il file è aperto in modalità *Output* o *Append*, il contenuto finale del buffer viene trasferito nel file sul disco.

Scrittura e lettura in un file sequenziale

La scrittura dei dati nel file sequenziale viene eseguita con la funzione **PrintLine**. La sintassi è:

PrintLine(*NumeroFile*, *Dato*)

- *NumeroFile* identifica il file sequenziale su cui scrivere, attraverso il numero assegnato ad esso con l'istruzione *FileOpen*;
- *Dato* indica la variabile contenente il dato da registrare nel file.

L'uso del numero di file con l'istruzione *PrintLine* è importante, perché in uno stesso programma possono essere aperti più file contemporaneamente.

Progetto 1

Creare un archivio con i nomi degli amici.

Si vuole creare un archivio su memoria di massa che contenga i nomi dei propri amici registrando i dati inseriti da tastiera.

Dati di input:
nomi degli amici

Dati di output:
archivio su disco.

Nome del progetto
ScriviAmici di tipo *Applicazione Windows Form*.

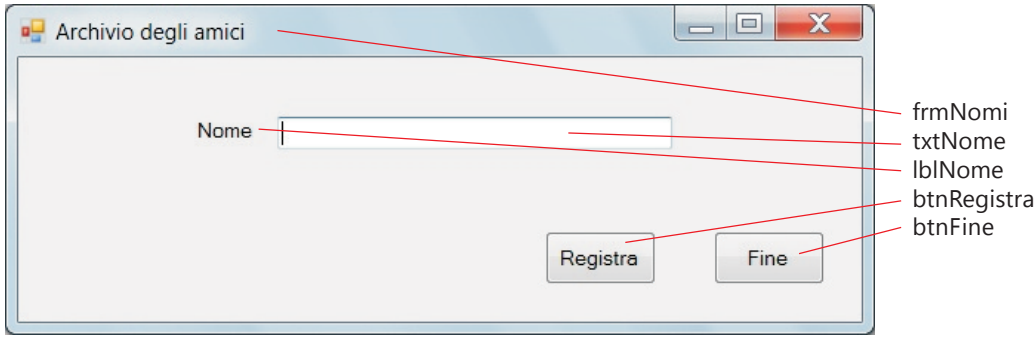
Disegno dell'interfaccia grafica
L'interfaccia è costituita da un form che contiene una casella di testo per l'inserimento del nome con la relativa etichetta di descrizione, il pulsante per registrare il nome sul file e il pulsante per chiudere il file e terminare il programma.

Classe	Proprietà dell'oggetto	
Form	Name	frmNomi
	Text	*Archivio di amici*
Label	Name	lblNome
	Text	*Nome*
TextBox	Name	txtNome
Button	Name	btnRegistra
	Text	*Registra*
Button	Name	btnFine
	Text	*Fine*

Gestione degli eventi

L'apertura del file è associata all'evento *Load* sul form. I nomi devono essere inseriti uno ad uno da tastiera e registrati su un file di tipo sequenziale ogni volta che si fa clic sul pulsante *Registra*: ogni riga del file contiene un nome di amico. L'archivio è aperto in scrittura (*OpenMode.Output*) e ciò comporta la cancellazione di un eventuale archivio preesistente con lo stesso nome e la scrittura dei dati inseriti a partire dall'inizio del file.

Dopo aver registrato il nome con l'istruzione *PrintLine*, la casella di testo viene svuotata (metodo *Clear*) e ad essa viene assegnata l'evidenziazione (metodo *Focus*) in modo che l'utente sia pronto per inserire il nome successivo. La chiusura del file con l'istruzione *FileClose* è associata all'evento *Click* sul pulsante *btnFine*.



Codice Visual Basic

```
   Private Sub frmNomi_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
      ' apre il file in scrittura
      FileOpen(1, "C:\Esercizi\Amici.dat", OpenMode.Output)
   End Sub

   Private Sub btnRegistra_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRegistra.Click
      PrintLine(1, txtNome.Text)  ' scrive il nome sul file
      txtNome.Clear() ' svuota la casella di testo
      txtNome.Focus() ' riassegna il focus alla casella
   End Sub

   Private Sub btnFine_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFine.Click
      FileClose(1) ' chiude il file
      End
   End Sub
```

Per controllare e visualizzare il contenuto di un file sequenziale si può utilizzare un programma di lettura.

In questo caso il file deve essere aperto in modalità Input:

FileOpen (*NumeroFile*, *NomePercorso*, **OpenMode.Input**)

In fase di lettura, i dati scritti tramite *PrintLine* vengono in genere letti dal file con l'istruzione **LineInput.**
La sintassi dell'istruzione è del tutto analoga a quella della corrispondente istruzione di scrittura:

dato = **LineInput**(*numerofile*)

Poiché nell'accesso sequenziale i dati registrati vengono letti dal file uno di seguito all'altro, a partire dall'inizio, nello stesso ordine con il quale erano stati scritti, occorre inserire all'interno del programma di lettura un controllo sul raggiungimento della fine del file. Tale controllo viene realizzato dalla **funzione EOF** (*End Of File*), che ha come argomento il numero del file e che assume il valore *True* appena viene raggiunta la fine del file.
Quindi il ciclo di lettura sequenziale di tutti i dati contenuti in un file può essere realizzato con una struttura *Do While* con il seguente schema generale:

```
Do While Not EOF(FileNum)
   Riga = LineInput(FileNum)
Loop
```

Si noti che, nel caso in cui il file sia vuoto, la funzione *EOF* assume subito il valore *True* e pertanto il ciclo di lettura non viene eseguito.

Progetto 2

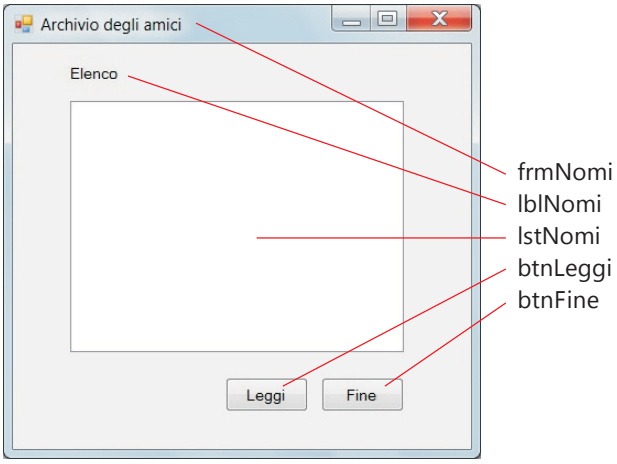
Visualizzare il contenuto dell'archivio degli amici.

Dati di input: archivio degli amici su disco
Dati di output: nomi degli amici in una lista sul video.

Nome del progetto
LeggiAmici di tipo *Applicazione Windows Form*.

Disegno dell'interfaccia grafica
Il form contiene una *ListBox* per visualizzare i nomi che vengono letti dal file con un'etichetta di intestazione. Impostando a *True* il valore della proprietà **Sorted** nel controllo *ListBox* si possono ottenere i nomi in ordine alfabetico, anziché in ordine di lettura dal file.

Classe	Proprietà dell'oggetto	
Form	Name	frmNomi
	Text	*Archivio di amici*
Label	Name	lblNomi
	Text	*Elenco*
ListBox	Name	lstNomi
	Sorted	True
Button	Name	btnLeggi
	Text	*Leggi*
Button	Name	btnFine
	Text	*Fine*



Gestione degli eventi
Il pulsante *Leggi* avvia l'operazione di lettura; il pulsante *Fine* chiude il file e termina il programma.
Per la visualizzazione del contenuto dell'archivio degli amici si procede utilizzando un'iterazione che comprende la lettura sequenziale di ogni riga del file e la sua visualizzazione in una lista. L'iterazione si arresta dopo che è stato letta l'ultima registrazione del file.
Il procedimento risolutivo è descritto dal seguente algoritmo.

Algoritmo
inizio
apri il file in lettura
esegui mentre il file non è finito
  leggi un nome dal file
  aggiungi il nome alla lista
ripeti
chiudi il file
fine

Codice Visual Basic

```
   Dim Nome As String
   Private Sub frmNomi_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
      ' apre il file in lettura
      FileOpen(1, "C:\Esercizi\Amici.dat", OpenMode.Input)
   End Sub

   Private Sub btnLeggi_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLeggi.Click
      Do While Not EOF(1) ' ripete fino alla fine del file
         Nome = LineInput(1) ' legge il dato nella variabile
         ' aggiunge nella lista il nome
         lstNomi.Items.Add(Nome)
      Loop
   End Sub

   Private Sub btnFine_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFine.Click
      FileClose(1) ' chiude il file
      End
   End Sub
```

Si possono aggiungere altri dati ad un archivio creato in precedenza specificando la modalità **Append**, al posto di *Output*, nell'apertura del file:

FileOpen (*NumeroFile*, *NomePercorso*, **OpenMode.Append**)

Per esempio, se si vogliono aggiungere altri nomi di amici all'archivio, si può usare un programma del tutto simile a quello usato per l'inserimento dei nomi.

Le modifiche riguardano la proprietà *Text* del pulsante di comando che attiva la registrazione (meglio usare *Aggiungi*, anziché *Registra*) e la specificazione della modalità di accesso, nella routine di gestione dell'evento *Load* del form, che diventa:

```
    Private Sub frmNomi_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        ' apre il file in scrittura
        FileOpen(1, "c:\Esercizi\Amici.dat", OpenMode.Append)
    End Sub
```

Dopo aver costruito il form per l'aggiunta di nuovi nomi, si può verificare, eseguendo di nuovo il programma di lettura del file, che i nomi già esistenti nell'archivio sono stati mantenuti.

Accesso diretto ai file di record

Nella gestione tradizionale degli archivi, oltre ai file sequenziali, si utilizzano gli archivi ad accesso diretto (*random*).
Nelle applicazioni gestionali l'uso di archivi su memoria di massa riguarda comunemente file formati da record aventi tutti la stessa struttura: si pensi agli archivi anagrafici di Clienti e Fornitori di un'azienda oppure agli archivi di Docenti e Studenti in una scuola, agli archivi dei Movimenti di contabilità oppure dei Movimenti di magazzino, ecc.
In tutti questi casi il programma di gestione degli archivi contiene dichiarazioni di strutture come la seguente, riferita alle informazioni anagrafiche del dipendente di un'azienda:

```
    Structure Persona
        Public Cognome As String
        Public Nome As String
        Public Stipendio As Double
        Public Funzione As String
    End Structure
```

Dopo aver definito la struttura, occorre dichiarare la variabile necessaria per l'elaborazione del record nel file:

```
Dim Dipendente As Persona
```

Si suppone che la *Matricola* identificativa del dipendente sia un numero intero che coincide con la posizione del record nel file. È quindi opportuno introdurre una variabile (che deve essere di tipo **Long**) per indicare il numero del record all'interno dell'archivio:

```
Dim Posizione As Long
```

Il file con accesso diretto deve essere aperto con modalità **Random**. Si noti che l'accesso di tipo random consente sia le operazioni di lettura che di scrittura.

La sintassi della funzione **FileOpen** per i file con accesso casuale è identica a quella già vista in precedenza per i file sequenziali: occorre solo cambiare la modalità di accesso specificando **OpenMode.Random**.

```
FileOpen(1, "C:\Esercizi\Anagrafe.dat", OpenMode.Random)
```

Anziché assegnare il valore 1 al primo parametro (numero del file), si può usare la funzione **FreeFile** che restituisce il primo numero di file disponibile da utilizzare con la funzione *FileOpen*.
La funzione *FreeFile* viene utilizzata quando è necessario fornire un numero di file e si desidera verificare che il numero di file non sia in uso. Questo modalità è utile soprattutto nelle applicazioni che prevedono l'uso di più file (anche con accessi diversi) che vengono aperti e chiusi più volte nel corso dell'applicazione.

```
Dim fileNum As Integer
fileNum = FreeFile()
FileOpen(fileNum, "C:\Esercizi\Anagrafe.dat", OpenMode.Random)
```

L'operazione di **scrittura** di un record su un file con accesso casuale è eseguita con la funzione **FilePut** seguendo la seguente sintassi:

FilePut (*NumeroFile*, *NomeRecord*, *Posizione*)

- *NumeroFile* è il numero del file all'interno del programma, assegnato con la funzione *FileOpen*
- *NomeRecord* è la variabile di tipo record che contiene i campi da scrivere sul file
- *Posizione* indica il numero del record dove inizia la nuova registrazione.

Con riferimento all'esempio di struttura precedente per i dipendenti, l'istruzione di scrittura diventa:

```
FilePut(1, Dipendente, Posizione)
```

L'operazione di **lettura** di un record da un file con accesso random è eseguita tramite l'istruzione **FileGet** avente la seguente sintassi generale, analoga a quella della funzione *FilePut*:

FileGet(*NumeroFile*, *NomeRecord*, *Posizione*)

Progetto 3

Creare un'applicazione per inserire nuovi dipendenti nell'archivio anagrafico di un'azienda e per visualizzare le informazioni di un dipendente di cui si fornisce la matricola.

Per il sottoprogramma di inserimento

Dati di input:
Matricola, Cognome, Nome, Stipendio e Funzione di un dipendente

Dati di output:
archivio anagrafico dei dipendenti dell'azienda.

Per il sottoprogramma di visualizzazione

Dati di input:
Matricola di un dipendente

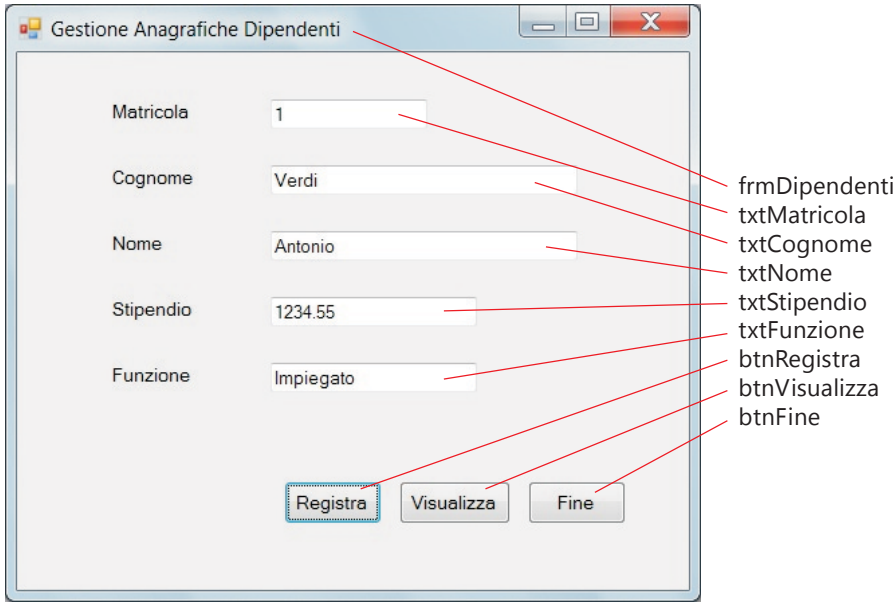
Dati di output:
Cognome, Nome, Stipendio e Funzione del dipendente

Nome del progetto
Dipendenti di tipo *Applicazione Windows Form*.

Disegno dell'interfaccia grafica
Il form contiene le caselle di testo per acquisire i valori da assegnare ai campi del record del dipendente. La casella di testo *txtMatricola* fornisce la posizione del dipendente nell'archivio. Ci sono anche tre pulsanti di comando: *Registra* per memorizzare i dati sul file, *Visualizza* per mostrare le informazioni del dipendente richiesto e *Fine* per uscire dall'applicazione.

Classe	Proprietà dell'oggetto	
Form	Name	frmDipendenti
	Text	*Gestione Anagrafiche Dipendenti*
TextBox	Name	txtMatricola
TextBox	Name	txtCognome
TextBox	Name	txtNome
TextBox	Name	txtStipendio
TextBox	Name	txtFunzione
Button	Name	btnRegistra
	Text	*Registra*
Button	Name	btnVisualizza
	Text	*Visualizza*
Button	Name	btnFine
	Text	*Fine*

Per semplicità, nella descrizione degli oggetti, sono state omesse le *Label* poste accanto alle caselle di testo.



Gestione degli eventi

L'operazione di apertura del file è eseguita all'inizio del programma e quindi è associata all'evento *Load* del form. Ciascun record è individuato dal numero di posizione che viene fornito con la casella di testo *txtMatricola*. Gli altri dati del dipendente vengono inseriti da tastiera e memorizzati nel file quando si fa clic sul pulsante *Registra*. Fornendo invece il numero di matricola e facendo poi clic sul pulsante *Visualizza*, il programma mostra i dati del dipendente, oppure le caselle vuote se non c'è alcun dipendente registrato con la matricola richiesta.

Nell'operazione di visualizzazione, quando l'utente fornisce un numero di matricola superiore al più alto numero tra quelli dei dipendenti già registrati, si ottiene un *errore di runtime* che blocca l'esecuzione del programma. La gestione di questa situazione di errore (*eccezione*) deve essere risolta come spiegato nel prossimo paragrafo.

Codice Visual Basic

```
Structure Persona
      Public Cognome As String
      Public Nome As String
      Public Stipendio As Double
      Public Funzione As String
End Structure

   Dim Dipendente As Persona
   Dim Posizione As Long

   Private Sub Form1_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
      ' apre il file
      FileOpen(1, "C:\Esercizi\Anagrafe.dat", OpenMode.Random)
   End Sub

   Private Sub btnRegistra_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRegistra.Click
      Posizione = Val(txtMatricola.Text)
      With Dipendente
         .Cognome = txtCognome.Text
         .Nome = txtNome.Text
         .Stipendio = Val(txtStipendio.Text)
         .Funzione = txtFunzione.Text
      End With
      ' scrive il record del dipendente
      FilePut(1, Dipendente, Posizione)
      ' prepara la maschera per un nuovo inserimento
      txtMatricola.Clear()
      txtCognome.Clear()
      txtNome.Clear()
      txtStipendio.Clear()
      txtFunzione.Clear()
      txtMatricola.Focus()
   End Sub
```

```
   Private Sub btnVisualizza_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnVisualizza.Click
      Posizione = Val(txtMatricola.Text)
      ' legge il record del dipendente
      FileGet(1, Dipendente, Posizione)
      ' prepara la maschera con i dati
      With Dipendente
         txtCognome.Text = .Cognome
         txtNome.Text = .Nome
         txtStipendio.Text = .Stipendio
         txtFunzione.Text = .Funzione
      End With
   End Sub

   Private Sub btnFine_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFine.Click
      FileClose(1) 'chiude il file
      End
   End Sub
```

Si noti che, richiamando con il pulsante *Visualizza* i dati di un dipendente già registrato e inserendo nuovi valori nelle caselle di testo, è possibile effettuare operazioni di aggiornamento dei dipendenti: si mantiene lo stesso numero di matricola e si fa clic sul pulsante *Registra*. Il nuovo record viene riscritto sopra al precedente.

Nell'accesso diretto ad un file è possibile usare anche la funzione **Seek**, che imposta la posizione della successiva operazione di lettura o scrittura. Quindi, se si usa la funzione *Seek*, non è più necessario specificare il terzo parametro delle istruzioni *FileGet* e *FilePut* indicante la posizione del record.

Per esempio, per leggere il quarto record del file utilizzato nell'esempio precedente, si può scrivere:

```
FileGet(1, Dipendente, 4)
```

oppure

```
Seek(1, 4)
FileGet(1, Dipendente)
```

L'istruzione *Seek* può essere utile per posizionarsi su un certo record e leggere il file da lì in poi. Per esempio se si vogliono leggere i dipendenti a partire dalla matricola 100, si può utilizzare un ciclo di lettura che parte dalla posizione 100 e termina quando viene raggiunta la fine del file.

```
Seek(1, 100)
Do While Not EOF(1)
   FileGet(1, Dipendente)
   . . . .
Loop
```

Lettura e scrittura di un file di testo

I metodi dell'oggetto **My.Computer.FileSystem** consentono di scrivere e leggere testo da un file con una sola istruzione del linguaggio Visual Basic.

Il metodo **WriteAllText** scrive un testo nel file e poi chiude il file. La sintassi generale è la seguente:

My.Computer.FileSystem.**WriteAllText**(*NomeFile*, *testo*, *append*)

Il primo parametro indica il nome del file, il secondo rappresenta la stringa o la casella di testo da registrare sul file e il terzo parametro, che può essere *True* o *False*, specifica se si deve aggiungere il testo (*True*) o sovrascrivere quello esistente (*False*); il valore predefinito è *False*. Se il file non esiste, viene creato.

Per scrivere dati che non sono di tipo testo (dati *binari*) si usa il metodo **WriteAllBytes** con una sintassi simile alla precedente, con la differenza che il secondo parametro deve essere un array di bytes anziché una stringa di testo.

La lettura dei dati si ottiene con il metodo **ReadAllText** con la seguente sintassi generale:

testo = My.Computer.FileSystem.**ReadAllText**(*NomeFile*)

Il corrispondente metodo per i dati di tipo binario è **ReadAllBytes**, che restituisce il contenuto del file in un array di bytes.

Progetto 4

Costruire l'applicazione che consente di creare un nuovo file di testo oppure di visualizzare il contenuto di un file di testo già esistente.

Le righe di testo sono scritte e lette all'interno di una casella di testo. L'utente può scegliere di creare un nuovo file con la registrazione di queste righe, dopo aver specificato il nome da assegnare al file, oppure può leggere il contenuto di un file già esistente che seleziona sfogliando le cartelle del disco.

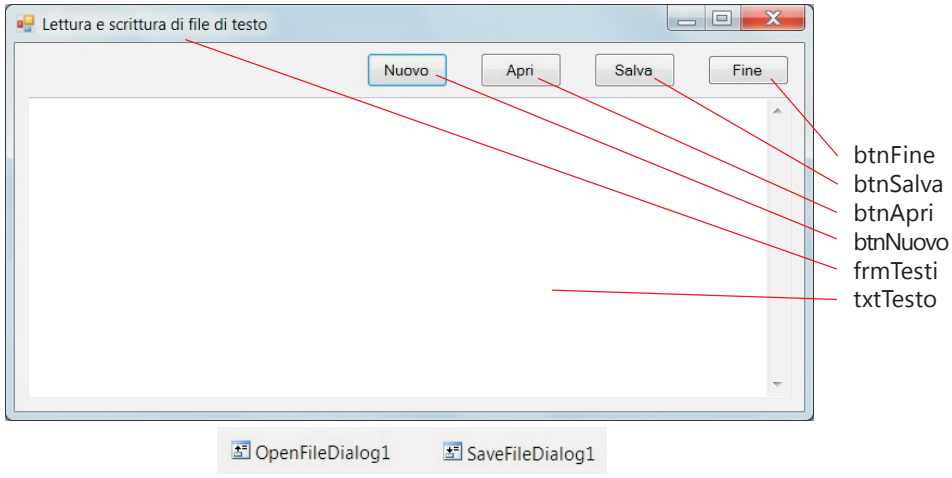
Nome del progetto

Testi di tipo *Applicazione Windows Form*.

Disegno dell'interfaccia grafica

Il form dell'applicazione contiene una casella di testo con la proprietà **Multiline** impostata a *True* per consentire l'inserimento e la visualizzazione di più righe. La proprietà **ScrollBars** è inoltre impostata a **Both** per visualizzare entrambe le barre di scorrimento quando il testo non può essere contenuto interamente nella casella di testo.
Il pulsante *Nuovo* svuota la casella di testo preparandola per un nuovo inserimento di testo.
I pulsanti *Salva* e *Apri* attivano le finestre di dialogo comuni **SaveFileDialog** e **OpenFileDialog**, che devono essere inserite nella finestra di progettazione.
Il pulsante *Fine* chiude l'applicazione.

Classe	Proprietà dell'oggetto	
Form	Name	frmTesti
	Text	*Lettura e scrittura di file di testo*
TextBox	Name	txtTesto
	Multiline	True
	ScrollBars	Both
Button	Name	btnNuovo
	Text	*Nuovo*
Button	Name	btnApri
	Text	*Apri*
Button	Name	btnSalva
	Text	*Salva*
Button	Name	btnFine
	Text	*Fine*



Gestione degli eventi

All'inizio la casella di testo è vuota e il titolo del form è "*Lettura e scrittura di file di testo*". Facendo clic sul pulsante *Nuovo*, la casella di testo viene svuotata e il titolo del form diventa "*Nuovo testo*".

L'evento *Click* sul pulsante *Apri* attiva la finestra di dialogo per la scelta del file da aprire: il nome del file viene scritto sulla barra del titolo della finestra e il contenuto del file viene visualizzato nella casella di testo.

L'evento *Click* sul pulsante *Salva* apre la finestra di dialogo per assegnare il nome al file: il nome del file viene scritto sulla barra del titolo della finestra e il contenuto della casella di testo viene scritto sul file. Se si assegna un nome di file che esiste già, viene richiesta la conferma per la riscrittura.

Il titolo nella barra della finestra è impostato con l'istruzione:

```
Me.Text = SaveFileDialog1.FileName
```

dove l'oggetto **Me** rappresenta il form stesso in esecuzione e la proprietà **FileName** contiene il nome assegnato al file nella finestra *Salva*.

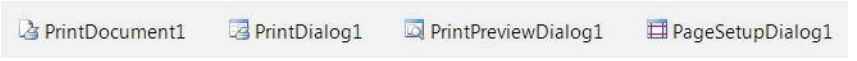
Codice Visual Basic

```
   Private Sub btnApri_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnApri.Click
   ' impostazioni della finestra di dialogo
   OpenFileDialog1.DefaultExt = "txt" ' estensione di default
   OpenFileDialog1.Filter = "File di testo|*.txt|Tutti i file|*.*"
   OpenFileDialog1.FileName = ""
   ' apre il file e lo visualizza nella casella di testo
   If OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK
   Then
      Me.Text = OpenFileDialog1.FileName
      txtTesto.Text =
My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName)
   End If
End Sub

   Private Sub btnNuovo_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNuovo.Click
      txtTesto.Clear()
      Me.Text = "Nuovo testo"
   End Sub

   Private Sub btnSalva_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSalva.Click
      ' impostazioni della finestra di dialogo
      SaveFileDialog1.DefaultExt = "*.txt"
      SaveFileDialog1.AddExtension = True
      SaveFileDialog1.FileName = ""
      SaveFileDialog1.Filter = "File di testo|*.txt|Tutti i file|*.*"
      ' salva il testo nel file
      If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK
      Then
         My.Computer.FileSystem.WriteAllText(SaveFileDialog1.FileName,
txtTesto.Text, False)
         Me.Text = SaveFileDialog1.FileName
      End If
   End Sub

   Private Sub btnFine_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFine.Click
      End
   End Sub
```

Le stampe

I risultati di un'elaborazione oppure i dati contenuti in un archivio possono essere riprodotti su carta secondo formati o modelli di stampa, indicati comunemente con il termine **report**.

La modalità più semplice per attivare una stampa consiste nell'usare l'oggetto **PrintDocument** di Visual Basic, che corrisponde al documento da inviare alla stampante predefinita nell'ambiente Windows, salvo dichiarazione di una stampante diversa.
L'operazione di stampa è attivata mediante il metodo **Print** applicato all'oggetto *PrintDocument*: questo metodo attiva l'evento **PrintPage** che si occupa della gestione effettiva dell'output sulla stampante secondo le impostazioni predefinite (di *default*) o quelle scelte dall'utente.
Le impostazioni possono riguardare per esempio le dimensioni del foglio di carta, la disposizione della stampa in orizzontale o verticale, oltre alla stampante da utilizzare tra quelle disponibili nel computer.
Gli oggetti per le operazioni di stampa sono le finestre di dialogo, già presentate nel Capitolo 6, raggruppate nella categoria **Stampa** della *Casella degli strumenti*: in fase di progettazione del form, l'oggetto non viene inserito all'interno del form, ma rimane posizionato nella parte inferiore della finestra di progettazione.



- **PageSetupDialog**

finestra di dialogo che consente agli utenti di gestire le impostazioni della pagina, compresi i margini e l'orientamento (corrisponde alla scelta **Imposta pagina** delle applicazioni Windows).

- **PrintPreviewDialog**

finestra di dialogo che consente di visualizzare un documento così come verrà stampato; contiene pulsanti per la stampa, l'ingrandimento, la visualizzazione di una o più pagine e la chiusura della finestra di dialogo (corrisponde alla scelta **Anteprima di stampa** delle applicazioni Windows).

- **PrintDialog**

finestra di dialogo che consente agli utenti di selezionare una stampante e scegliere le parti del documento da stampare (corrisponde alla scelta **Stampa** delle applicazioni Windows).

È disponibile anche l'oggetto **PrintPreviewControl** che consente di inserire nel progetto una finestra di anteprima personalizzata, anziché utilizzare la finestra standard fornita da *PrintPreviewDialog*.

Come già visto, per aprire una finestra di dialogo, occorre utilizzare nel codice il metodo **ShowDialog** applicato alla finestra.

Per esempio:

```
PrintPreviewDialog1.ShowDialog()
```

apre la finestra di dialogo predefinita per l'anteprima di stampa.

È possibile controllare i pulsanti premuti dall'utente all'interno della finestra di dialogo attraverso l'insieme delle costanti **DialogResult**.

Per esempio, la seguente struttura *If* manda in esecuzione il metodo *Print* per l'oggetto *PrintDocument1* nel caso in cui l'utente abbia premuto il pulsante *OK* all'interno della finestra di dialogo per la stampa:

```
If PrintDialog1.ShowDialog = DialogResult.OK Then
   PrintDocument1.Print()
End If
```

Tutte le finestre di dialogo devono far riferimento all'oggetto *PrintDocument1*: questa impostazione è realizzata assegnando l'oggetto alla proprietà **Document** delle finestre; per esempio:

```
PageSetupDialog1.Document = PrintDocument1
```

La stampa del testo è realizzata con il metodo **Graphics.DrawString**, che viene eseguito nella subroutine di gestione dell'evento *PrintPage*: questo evento viene automaticamente attivato quando si esegue il metodo *Print* sull'oggetto di tipo *PrintDocument*, oppure quando si richiede l'anteprima di stampa con la finestra di dialogo *PrintPreviewDialog*.

Il metodo *Graphics.DrawString* ha quattro parametri principali:
- la stringa di testo da inviare alla stampante
- il tipo di font da utilizzare
- il pennello di colore per la stampa
- le coordinate del punto di partenza della stampa sul foglio a partire dall'angolo in alto a sinistra del foglio (numeri interi di tipo *Single*).

Il pennello di colore è rappresentato dall'insieme **Brushes** specificando come proprietà il nome del colore, per esempio il colore nero si rappresenta con *Brushes.Black*.
La seguente istruzione stampa su carta la stringa *testo*, utilizzando il *font* specificato come secondo parametro, in colore nero (*Black*), a partire dal punto di coordinate *x*, *y*:

```
ev.Graphics.DrawString(testo, font, Brushes.Black, x, y)
```

Più precisamente la variabile *ev* fornisce i dati all'evento *PrintPage*; l'oggetto *ev.Graphics* attraverso i suoi metodi disegna la pagina sulla stampante.
Altri metodi della classe *Graphics* consentono la stampa di immagini, linee o forme, per esempio:

```
ev.Graphics.DrawImage(Image.FromFile("C:\images\foto1.jpg"), 20, 50)
```

per stampare l'immagine *foto1.jpg* a partire dalla posizione (20, 50)

```
ev.Graphics.DrawLine(Pens.Red, 100, 150, 300, 400)
```

per tracciare una linea di colore rosso (*Pens.Red*) dal punto (100,150) al punto (300,400)

```
ev.Graphics.DrawRectangle(Pens.Blue, 100, 200, 250, 300)
```

per disegnare un rettangolo con il contorno blu dal punto (100, 200) al punto (250, 300).

Le impostazioni di default per la stampa sono contenute nella proprietà **DefaultPageSettings** dell'oggetto *PrintDocument*. Se queste impostazioni vengono cambiate dall'utente attraverso la finestra di dialogo *PageSetupDialog*, possono essere poi riassegnate come nuovi valori di default, con la seguente istruzione:

```
PrintDocument1.DefaultPageSettings = PageSetupDialog1.PageSettings
```

Per utilizzare gli eventi e i metodi riguardanti le operazioni di stampa, occorre collegare all'applicazione le apposite librerie inserendo, all'inizio nella finestra del codice, la seguente dichiarazione:

```
Imports System.Drawing.Printing
```

È buona norma verificare l'anteprima di stampa prima di mandare il documento in stampa. Inoltre, in fase di sviluppo dell'applicazione, per fare prove di esecuzione, può essere utile scegliere una stampante logica, per esempio la stampante che produce file **pdf**, anziché utilizzare i fogli di carta di una stampante fisica.

Stampa del contenuto di un file di testo

Gli oggetti per la stampa possono essere utilizzati nell'applicazione che è stata presentata nel Progetto 4 per la gestione di un file di testo.

Progetto 5

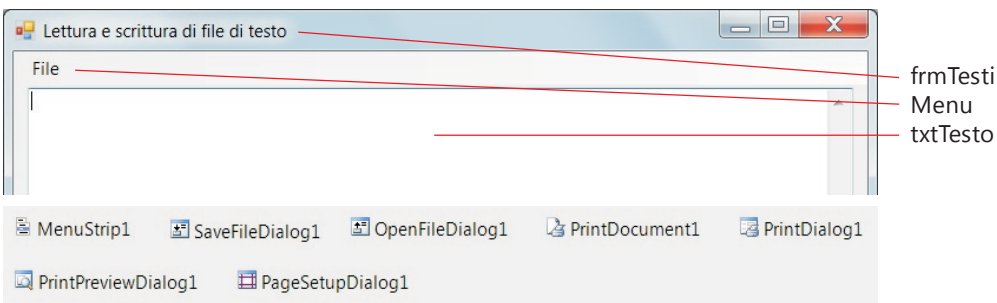
Stampare su carta il contenuto di un file di testo di cui viene fornito il nome.

In questo progetto vengono presentate le modifiche e le aggiunte per ampliare le funzionalità dell'applicazione del Progetto 4 con l'aggiunta della stampa del file di testo. L'interfaccia viene anche rielaborata organizzando le scelte con un **menu**, anziché usare i pulsanti di comando.

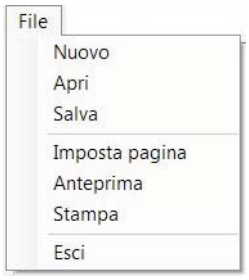
Nome del progetto
Testi2 di tipo *Applicazione Windows Form*.

Disegno dell'interfaccia grafica
Utilizzando il menu a tendina, al posto dei pulsanti di comando, gli elementi dell'interfaccia sono: la casella di testo e il **MenuStrip**. Nel progetto devono essere inseriti anche i componenti per il menu e per le finestre di dialogo, che saranno posizionate nella parte inferiore della finestra di progettazione.

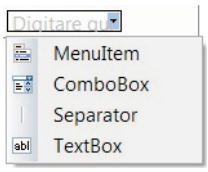
Classe	Proprietà dell'oggetto	
Form	Name	frmTesti
	Text	*Lettura e scrittura di file di testo*
TextBox	Name	txtTesto
	Multiline	True
	ScrollBars	Both
MenuStrip	Name	Menu



La voce principale del menu è *File* e ha le voci secondarie illustrate in figura:



Per inserire un separatore orizzontale tra le voci di menu, si deve fare clic sulla freccia verso il basso della casella che contiene la frase "*Digitare qui*" e selezionare **Separator**.



Gestione degli eventi

Le subroutine di gestione degli eventi sono analoghe a quelle utilizzate nel Progetto 4 per *Nuovo*, *Apri*, *Salva*.

Per inserire il codice, basta copiare il codice associato a ciascun pulsante del Progetto 4, fare doppio clic sulla voce corrispondente del menu e incollare le righe di codice. Per esempio la gestione dell'evento *Click* sulla voce di menu *Nuovo* (*NuovoToolStripMenuItem_ Click*) è rappresentata dalla seguente subroutine che contiene le stesse istruzioni della subroutine del Progetto 4:

```
   Private Sub NuovoToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles NuovoToolStripMenuItem.Click
      txtTesto.Clear()
      Me.Text = "Nuovo testo"
   End Sub
```

Per l'opzione *Stampa* si ricordi di inserire anche la subroutine *PrintDocument1_PrintPage* che risulta modificata come indicato di seguito.

L'opzione *Esci* del menu corrisponde all'istruzione *End*.

Anche per questo progetto si deve inserire, come prima riga nella finestra del codice, la dichiarazione:

```
Imports System.Drawing.Printing
```

Codice Visual Basic

```
   Private Sub PrintDocument1_PrintPage(ByVal sender As Object, ByVal ev
As PrintPageEventArgs) Handles PrintDocument1.PrintPage
     Dim x, y As Single
     ' definisce i margini come coordinate di partenza
     x = PrintDocument1.DefaultPageSettings.Margins.Left
     y = PrintDocument1.DefaultPageSettings.Margins.Top
     ' stampa testo
     ev.Graphics.DrawString(txtTesto.Text, txtTesto.Font, Brushes.Black, x, y)
   End Sub
```