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

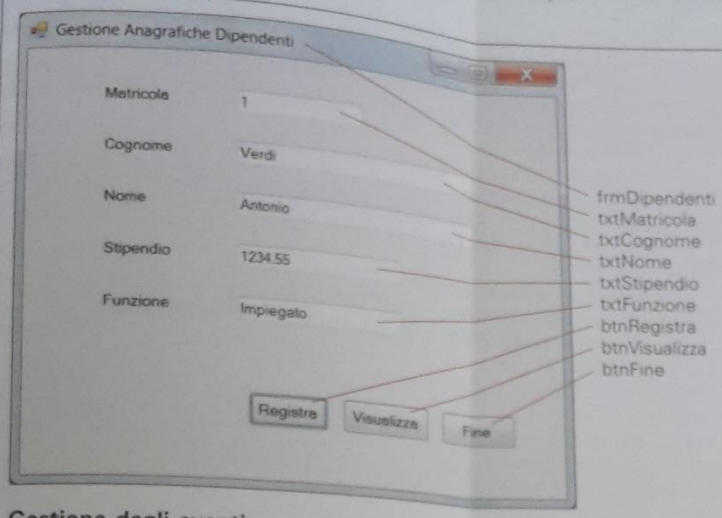
Dipendenti di tipo *Applicazione Windows Form*.

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
```

5 La gestione delle eccezioni

Un'applicazione software deve prevedere tutti i casi che si possono verificare durante l'esecuzione e fornire all'utente messaggi di avvertimento o di errore quando si verificano situazioni anomale dovute ad errati inserimenti di dati da parte dell'utente oppure ad errori del sistema. Questi tipi di errore si chiamano comunemente **errori di runtime**, cioè errori che si verificano durante l'esecuzione del programma.
Per costruire un'interfaccia amichevole ed efficace, è opportuno che il programmatore preveda all'interno del programma le routine che siano in grado di intercettare le situazioni di errore fornendo all'utente opportuni messaggi e vie di uscita.
Nell'attività di programmazione gli errori, che possono provocare un'interruzione anomala del programma, vengono indicati comunemente con il termine **eccezioni**.

Il linguaggio Visual Basic prevede la gestione strutturata delle eccezioni, racchiudendo gruppi di istruzioni che possono generare errori durante l'esecuzione e specificando le attività da svolgere quando l'eccezione viene rilevata e intercettata.
La gestione strutturata è rappresentata dall'istruzione

```
Try
  ...
Catch
  ...
Finally
  ...
End Try
```

La parola **Try** indica l'inizio della struttura di gestione dell'eccezione ed è seguita dal blocco delle istruzioni sottoposte al controllo perché possono generare un errore di esecuzione.
Dopo **Catch** viene inserito il codice che deve essere eseguito quando le istruzioni del blocco *Try* provocano un'interruzione dell'esecuzione perché si è verificato un errore.

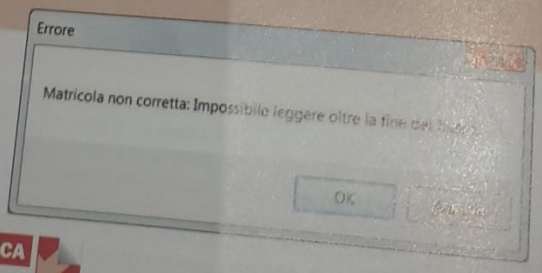
Se si vuole che il programma esegua altre istruzioni prima di uscire dalla struttura di controllo delle eccezioni, queste possono essere scritte dopo la parola **Finally**. Questa parte può anche non esserci.
Infine **End Try** chiude la struttura.

Il codice di gestione delle eccezioni utilizza un oggetto di tipo **Exception**, che consente di ottenere le informazioni sull'eccezione che si è verificata.
Le proprietà di *Exception* facilitano l'identificazione della posizione del codice, del tipo e della causa delle eccezioni:

- la proprietà **StackTrace** elenca i metodi che hanno provocato l'eccezione;
- la proprietà **Message** restituisce un messaggio di testo che descrive l'errore;
- la proprietà **Source** è una stringa che contiene il nome dell'oggetto che ha provocato l'errore.

Per esempio, nel Progetto 3 presentato in precedenza, si verifica un'eccezione quando l'utente chiede di visualizzare un dipendente fornendo un numero di matricola che corrispondente a una posizione oltre la fine del file.
La gestione dell'errore può essere realizzata modificando il sottoprogramma *Visualizza* e introducendo una struttura *Try ... Catch ... End Try* per intercettare l'eccezione e fornire un messaggio all'utente.

```
Private Sub btnVisualizza_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnVisualizza.Click
        Posizione = Val(txtMatricola.Text)
        Try
            ' 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
        Catch ex As Exception
            MessageBox.Show("Matricola non corretta: "
& ex.Message, "Errore", MessageBoxButtons.OKCancel)
        Finally
            txtMatricola.Focus()
        End Try
End Sub
```



AUTOVERIFICA
Domande da 3 a 8 pag. 508-509
Problemi da 1 a 11 pag. 511