# Quando usare file di dati

- Per salvare dati da un'esecuzione di un'applicazione alla successiva
- Da usare quando c'è soltanto una piccola quantità di dati, tale da non giustificare l'uso di un database
- Da usare per applicazioni Windows, poiché le norme di sicurezza per Internet e reti intranet non consentono l'accesso a file su disco

# File di dati

- Dati memorizzati in file su un'unità disco
- File ==> Raccolta completa di dati
- Record ==> Righe o linee, una per entità
- Campi ==> Elementi contenuti in una linea
- Di norma memorizzati in modo organizzato
  - Ordinati secondo uno dei campi
  - Campo chiave, campo con valore unico per ogni record

# File di dati d'esempio

| Cognome | Nome | Telefono |
|---|---|---|
| Weldon | Lynne | 803-649-9999 |
| Buckner | Jim | 803-652-1111 |
| Thumb | Tom | 803-593-1234 |

**Record**

**Campi**

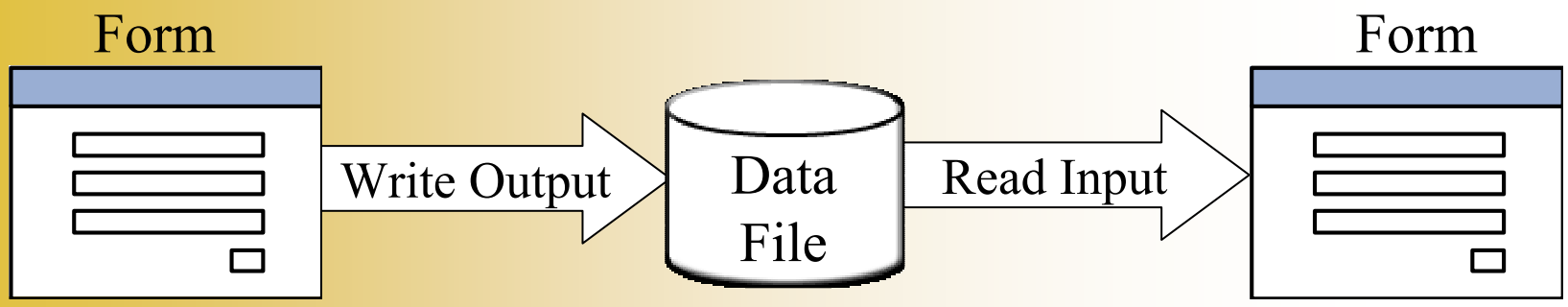
# Gestione dei file per mezzo di flussi (stream)

- Un **flusso (stream)** si occupa del trasferimento di una serie di byte da una posizione ad un'altra
- I flussi sono oggetti con proprietà e metodi
- Si trova nello spazio dei nomi System.IO
- I progetti che gestiscono file devono contenere un'istruzione Imports prima dell'istruzione che dichiara la classe del form (Public Class …):

**Imports System.IO**

# Input/Output (I/O)

- Leggere e scrivere dati in un file su disco
  - Leggere = **Input**
  - Scrivere = **Output**



# Scrittura dei file di dati

Per scrivere in un file di testo dei dati:

- Si dichiari un nuovo oggetto **StreamWriter**
  - Si indichi anche il nome del file di dati
- Si usi il metodo **WriteLine** di StreamWriter
  - Tale metodo copia i dati in un buffer in memoria prima che questi vengano scritti effettivamente su disco
- Al termine, si chiami il metodo **Close** di StreamWriter
  - Trasferisce i dati rimanenti dal buffer al file e rilascia le risorse di sistema usate dal flusso

# Oggetti StreamWriter

- Dichiarare un nuovo oggetto StreamWriter *apre* il file
- Se il file non esiste ne viene creato uno nuovo
- Si dichiari l'oggetto StreamWriter o nella sezione dichiarazioni o in una procedura

# Istanziare un oggetto StreamWriter

- Forma generale
  - La posizione di default del file è la cartella bin relativa al progetto corrente
  - Si può specificare il percorso completo del file

Dim *NomeOggetto* As New **StreamWriter**(*"NomeFile"*)

OPPURE

Dim *NomeOggetto* As **StreamWriter**

*NomeOggetto* = New **StreamWriter**(*"NomeFile"*)

# Esempi di dichiarazione di un oggetto StreamWriter

```
Dim datPhone As New StreamWriter("Phone.txt")
Dim datNames As New StreamWriter("C:\MyFiles\Names.txt")
Dim datLogFile As New StreamWriter("Logfile.txt", True)
```

**True = Aggiungi i dati al file esistente**
**False (o niente) = Sovrascrivi il file se esistente**

# Metodi Write e WriteLine

- Metodo Write
  - Inserisce i campi consecutivamente nel file senza alcun delimitatore (separatore)
- Metodo WriteLine
  - Inserisce ogni campo seguito da un Enter, cioè un ritorno a capo
  - Usato nel seguito in modo da facilitare il recupero dei dati

# Forma generale di WriteLine

- L'argomento *DatoDaScrivere* può essere una stringa o di tipo numerico
- In ogni caso, WriteLine converte qualsiasi valore numerico in una stringa e scrive quest'ultima nel file

*NomeFlusso*.**WriteLine**(*DatoDaScrivere*)

# Esempi di WriteLine

```
datPhone.WriteLine(txtName.Text)
datPhone.WriteLine(txtPhone.Text)

datNames.WriteLine("Sammy")

datBankBalance.WriteLine(decBalance.ToString)
```

# Chiudere un file di output

- Si usi il metodo **Close** di StreamWriter
- Tale metodo termina di scrivere sul disco tutti i dati contenuti nel buffer del flusso e rilascia le risorse di sistema
- Di solito codificato prima dell'uscita dall'applicazione

# Metodo Close

- Forma generale

  *NomeOggetto*.**Close**( )

- Esempi

  *datPhone*.**Close**( )

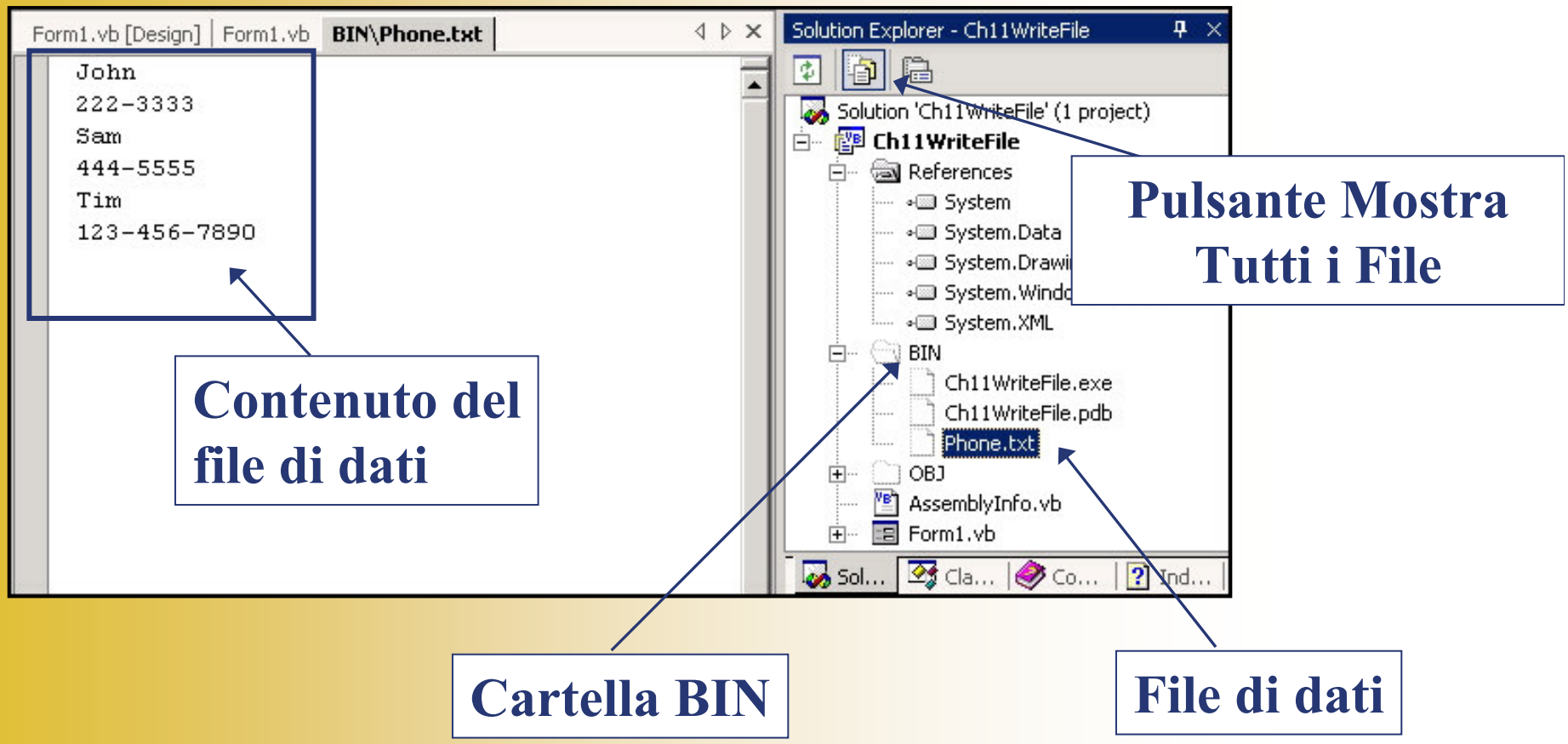
  *datNames*.**Close**( )

  *Me*.**Close**( )

# Vedere il contenuto del file

- Tramite un editor di testo come Notepad
- Per mezzo della finestra dell'editor IDE di Visual Studio
  - In Esplora Soluzioni
    1. Selezionare il Nome del Progetto
    2. Selezionare il pulsante Mostra Tutti i File per visualizzare la cartella BIN
    3. Selezionare il file da visualizzare nella finestra dell'editor

# Vedere il contenuto del file (cont.)



# Lettura dei file

Per leggere da un file di testo dei dati:

- Si dichiari un nuovo oggetto **StreamReader**
  - Si indichi anche il nome del file di dati, il quale verrà aperto quando l'oggetto StreamReader viene creato.
- Si usi il metodo **ReadLine** dello StreamReader per leggere i dati, eventualmente per mezzo di un ciclo
- Si chiami il metodo **Close** dello StreamReader per chiudere il file

# Oggetti StreamReader

- Dichiarare un nuovo oggetto StreamReader apre il file
- Il file deve esistere nella posizione specificata, altrimenti viene generato un errore
- Si dichiari l'oggetto StreamReader soltanto in una procedura, in modo da poterlo racchiudere in un blocco Try/Catch per la gestione degli errori

# Istanziare un oggetto StreamReader

Dim *NomeOggetto* As New **StreamReader**(*"NomeFile"*)

OPPURE

Dim *NomeOggetto* As **StreamReader**

*NomeOggetto* = New **StreamReader** (*"NomeFile"*)

# Esempio dichiarazione di un oggetto StreamReader

```
Try
    Dim datPhone As New StreamReader("Phone.txt")
Catch
    MessageBox.Show("Il file non esiste")
End Try
```

NB: Se il file da leggere non esiste, *si informi l'utente*

# Metodo ReadLine

- Usato per leggere dati precedentemente salvati
- Ogni volta che viene eseguito legge la linea di dati successiva
- Si assegni sempre il valore letto ad una posizione di memoria, quale un'etichetta, una casella di testo o una variabile stringa

# Forma generale ed esempi di ReadLine

- Forma generale

*NomeFlusso*.**ReadLine**( )

- Esempi

*txtName.Text* = *datPhone*.**ReadLine**( )

*lblPhone.Text* = *datPhone*.**ReadLine**( )

*strName* = *datNames*.**ReadLine**( )

# Controllare la fine del file

- Si usi il metodo **Peek** di StreamReader
- Peek esamina l'elemento successivo *senza* realmente leggerlo
- Se viene rilevata una posizione oltre l'ultimo elemento, il valore restituito è -1
- Si codifichi un test che esegua Peek e controlli che il valore restituito sia diverso da –1 prima di leggere il file

# Esempio di lettura di un file

Sia *reader* il nome dello StreamReader:

```
Do Until reader.Peek=-1
    MessageBox.Show(reader.ReadLine)
Loop
```

# Considerazioni sulla lettura e sulla scrittura di file

- Usando WriteLine viene scritto un unico elemento per ogni linea
- Usando ReadLine viene letto un unico elemento per ogni linea
- Si deve essere sicuri di leggere gli elementi esattamente nello stesso ordine nel quale erano stati scritti, al fine di ottenere dei dati validi

# Altri metodi per la lettura

- Il metodo

  **Read(*numero caratteri*)**

  legge il numero di caratteri specificato

- Il metodo **ReadToEnd** legge tutto il contenuto del file

# Finestre dialogo comuni di file

- Tutti i precedenti esempi di file di dati prevedevano che i nomi ed i percorsi dei file venissero scritti direttamente nel codice
- Può essere preferibile consentire all'utente di localizzare il file da aprire in fase di esecuzione
- Si usino i controlli finestre di dialogo comune **OpenFileDialog** e **SaveFileDialog** per visualizzare una finestra di dialogo standard Apri File o Salva File di Windows
- Si usi poi la proprietà **FileName** del controllo per aprire il file selezionato

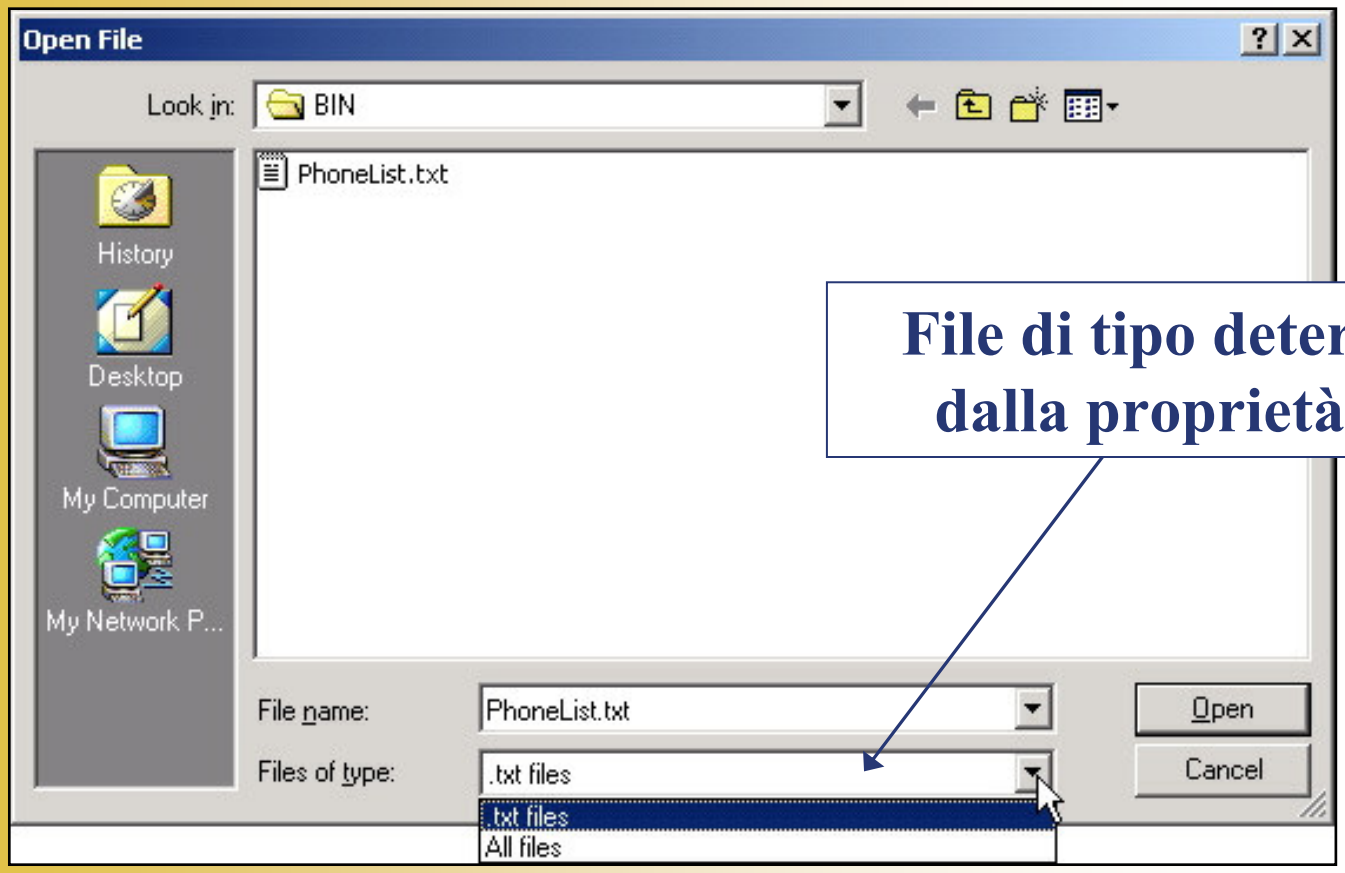
# Alcune proprietà del controllo OpenFileDialog

- **Name**: usare di norma il prefisso **dlg**
- **FileName**: sarà caricata con il valore selezionato dall'utente in esecuzione o con la stringa vuota qualora l'operazione venga annullata
- **Filter**: permette di indicare il filtro per la selezione del tipo di file; ad esempio, inserendo la stringa *"File di testo (*.txt)|*.txt"* verranno filtrati soltanto i file di suffisso *txt*.
- **InitialDirectory**: consente di indicare la cartella sulla quale si apre la finestra di dialogo ed è una proprietà da impostare solitamente nel codice

# Visualizzare una finestra di dialogo Apri File

- Aggiungere un controllo OpenFileDialog al form
  - Compare nel pannello componenti
- In fase di programmazione predisporre i valori iniziali delle sue proprietà eccetto InitialDirectory e FileName
- Nel codice impostare **InitialDirectory** ad **Application.StartUpPath**, che indicherà automaticamente la cartella dove si trova il programma.

# Dialogo Apri File




**File di tipo determinato dalla proprietà Filter**

# Esempio di dialogo Apri File - mnuFileOpen_Click

```
Dim dgrResult As DialogResult

dlgOpen.InitialDirectory=Application.StartupPath
dgrResult=dlgOpen.ShowDialog( )

' Si controlla se l'utente ha fatto click su Cancel
If dgrResult < > DialogResult.Cancel Then
    datPhone=New StreamWriter(dlgOpen.FileName)
End If
```

# Controllare se un file è stato aperto correttamente

- Se StreamWriter o StreamReader non sono stati istanziati, il file non è stato aperto
- Si usi la parola chiave **Nothing** per verificare l'istanziazione
- Un oggetto che non è stato istanziato ha un valore di Nothing

# Usare la parola chiave Nothing

```
' Si verifica se datPhone è stato istanziato
' Si verifica se il file è aperto

If Not datPhone Is Nothing Then
    datPhone.Close()
End If
```

# Evento Load del Form

- Ha luogo prima che un form venga visualizzato per la prima volta
- La relativa procedura di evento è il luogo indicato per caricare da file eventuali dati necessari al programma fin dall'inizio

# Evento FormClosing del Form

- La relativa procedura di evento è il miglior luogo per includere il codice da eseguire prima del termine del progetto
- Si codifichi in questa procedura la richiesta all'utente di salvataggio su file di eventuali dati
- Qualora l'utente non abbia modificato i dati dopo l'ultimo salvataggio, la richiesta NON dovrebbe aver luogo (si usi una variabile booleana a livello di modulo per tenere traccia di ciò)