Corso sul linguaggio C++

Modulo 6

1 - Struttura file testo

Prerequisiti

- Corso di programmazione base
- Memorie secondarie

Argomenti

- Una gerarchia per i dati
- Il file
- File di testo
- File binari
- Le classi di input/output
- I file in C++
- Operazioni sui file
- I file di testo
- Lettura da file di testo
- Scrittura su file di testo
- Accodamento su file di testo
- File da linea di comando
- Esempi
  - Gestire un elenco di nominativi
  - Gestire un file numerico

Introduzione

In questa Unità vediamo gli strumenti per l'utilizzo dei **file** in in C++.

Con i file è il programma è in grado di *comunicare i dati con il mondo esterno attraverso le periferiche*, cioè:

- i dispositivi standard di Input/Output (tastiera e video);
- l'uso delle memorie di massa (disco).

Una gerarchia per i dati

- **Bit** – la più piccola unità di informazione (valore: `0` o `1`)
- **Byte** – 8 bit (unità di indirizzamento della memoria)
  - dimensione di un carattere (cifre decimali, lettere e simboli speciali)
- **Variabile** – (nome simbolico contenente un dato);
  - dati semplici (intero, reale, logico ,booleano, carattere, stringa)
- **Array** – gruppo omogeneo di dati *correlati* ad 1 o più dimensioni
  - array di numeri, array di *struct* (tabella);
- **Record** (*struct*) – gruppo di campi *correlati*
  - scheda anagrafica (cognome, nome, indirizzo, codice fiscale)
- **File** – gruppo di record ***correlati***
  - archivio anagrafico, archivio articoli di magazzino
- **Database** – gruppo di fle ***correlati***
  - gestione integrata di un'attività commerciale, industriale,…

Il file

Il **file** è l'unità **logica** di memorizzazione dei dati su **memoria di massa**.

- Consente una memorizzazione **persistente** dei dati, non limitata dalle dimensioni della memoria centrale.
- Generalmente un file è una **sequenza** di componenti omogenee (*record logici*)
- I file sono gestiti dal *Sistema Operativo* sottostante e possono essere
  - **File di testo** (trattati in questa Unità)
  - **File binari**: (trattati nella successiva Unità)

File di testo

I **file di testo**

- sono costituiti da caratteri ed organizzati in linee (ciascuna terminata da '\n'), leggibili e portabili, ma lenti;
- ogni linea è terminata dal carattere di fine linea **newline** (carattere '\n');
- l'unità logica di riferimento può essere il **singolo carattere** oppure la **singola linea**

Egregio Sig. Rossi **\n**

Con la presente, La ringraziamo**\n** per il sollecito pagamento degli**\n** arretrati.**\n\n**

Gradisca cordiali saluti.

File binari

I **file binari**

- sono costituiti da sequenze di byte, organizzati in record logici
- non sono leggibili, non sempre sono portabili, ma sono veloci nell'I/O

Ad es. archivi di **struct**

File sequenziale

Rossi	Verdi	Bianchi	Celestini
Mario	Antonio	Elena	Anna
v. Po, 43	v. Piave, 3	v. Arno, 1	v. Tirso, 5
Torino	Napoli	Firenze	Cagliari
Record 1	**Record 2**	**Record 3**	**Record 4**

Le classi di input/output

In C++ il flusso dei dati tra programma e periferiche è gestito da *librerie di classi* dette **stream**, che possono essere immaginate come flussi di dati tra un *produttore* (sorgente) ed un *consumatore* (destinatario).

Uno **stream di input** è un flusso che ha come sorgente una periferica esterna e come destinatario la memoria (ad es. *tastiera* → *memoria*)

Uno **stream di output** è un flusso è un flusso che ha come sorgente una periferica esterna e come destinatario (ad es. *memoria* → *video*)

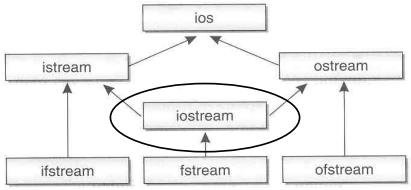
Uno **stream di input/output** è un flusso che ha come sorgente e destinatario una periferica esterna (ad es. *disco* → *disco*)

Le classi di input/output

La gerarchia delle classi di I/O è rappresentata in figura. Dalla classe **ios** derivano le classi:

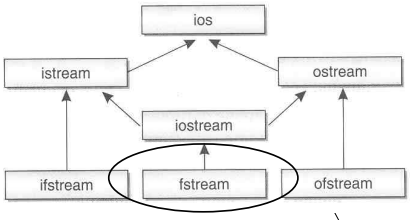
- **istream**, contenente gli oggetti per trattare *l'input* (es. **cin**)
- **ostream**, che contiene oggetti per trattare *l'output*.(es. **cout**)

La classe **iostream** deriva da **istream** e **ostream** ed eredita tutte le caratteristiche per trattare gli stream standard di I/O (**tastiera e video**)



Le classi di input/output

Le classi **fstream** (derivata da **iostream**) e **ifstream** e **ofstream** consentono di usare gli *stream* collegati ai file registrati su **memoria di massa**



I file in C++

I file in C++ sono **organizzati** in modo *sequenziale* con possibilità di **accesso**:

- **sequenziale** – reperimento di un record mediante scansione in sequenza di tutti i record precedenti
  - unico metodo possibile se l'organizzazione fisica è su nastro magnetico
  - sempre possibile se il file è su disco
- **diretto** – reperimento di un record in base alla sua posizione logica (tipico metodo utilizzato per file su disco)

Operazioni sui file

Le principali **operazioni logiche** sui file sono:

- Apertura file
  - in lettura
  - in scrittura
  - in accodamento
- Lettura da file
- Scrittura su file
- Chiusura del file
- Test su file

Operazioni sui file

Apertura

L'apertura di un file in C++ può essere fatta nei modi seguenti:

- Apertura file in **lettura**
- Apertura file in **scrittura**
- Apertura file in **accodamento**
- Apertura in **lettura/scrittura**

Operazioni sui file
Apertura in lettura

L'apertura in **lettura** di uno *stream di testo* in C++ può essere fatta mediante oggetti di classe **ifstream** nei modi seguenti:

ifstream *nomestream* (*nomefile*);	Apre in lettura *nomestream*
ifstream *nomestream*; *nomestream*.**open**(*nomefile*);	Istanzia l'oggetto *nomestream* Apre *nomestream* in lettura

Esempi:

1. **ifstream** textfile ("testo.txt");
2. **ifstream** textfile;
   textfile**.open** ("testo.txt");

Operazioni sui file
Apertura in scrittura

L'apertura in **scrittura** di uno stream in C++ può essere fatta mediante oggetti di classe **ofstream** nei modi seguenti:

ofstream *nomestream* (*nomefile*);	Apre in scrittura *nomestream*
ofstream *nomestream*; *nomestream*.**open**(*nomefile*);	Istanzia l'oggetto *nomestream* Apre *nomestream* in scrittura

Esempi:

1. **ofstream** textfile ("testo.txt");
2. **ofstream** textfile;
   textfile**.open** ("testo.dat");

Quando si apre in scrittura un file, viene *cancellato tutto il suo contenuto*

Operazioni sui file
Apertura in accodamento

L'apertura in **accodamento** di uno stream in C++ può essere fatta su oggetti di classe **ofstream** nei modi seguenti:

ofstream *nomestream*; *nomestream*.**open**(nomefile, **ios::app**);	Istanzia l'oggetto *nomestream* Apre *nomestream* in accodamento

Esempi:

1. **ofstream** testo;
   testo**.open** ("nomi.txt", **ios::app**);

Quando si apre uno stream in accodamento, i dati scritti *vengono aggiunti in coda a quelli esistenti.*

Operazioni sui file
Apertura in lettura/scrittura

L'apertura in **lettura/scrittura** di uno stream in C++ può essere fatta mediante oggetti di classe **fstream** nei modi seguenti:

fstream *nomestream*; *nomestream*.**open**(nomefile, **ios::in | ios::out**);	Istanzia l'oggetto *nomestream* Apre *nomestream* in lettura/scrittura (**testo**)

Esempi:

1. **fstream** dati;
   dati**.open** ("numeri", **ios::in | ios::out**);

Quando si apre un file in lettura e scrittura, è *possibile sia leggere che scrivere dati su esso.*

Operazioni sui file
Apertura: considerazioni

Quindi:

1. se lo stream deve essere aperto in lettura si deve istanziare la classe **ifstream** o **fstream**;
2. se lo stream deve essere aperto in scrittura o in accodamento si deve istanziare la classe **ofstream** o **fstream**;
3. se lo stream deve essere aperto in lettura/scrittura va istanziato dalla classe **fstream**;
4. se si usa la classe **fstream**, le modalità di apertura (lettura, scrittura, accodamento, binario) *devono essere esplicitate* tramite la costante **ios**.

Operazioni sui file
Lettura

L'operazione di **lettura** da file si esegue come indicato:

nomestream >> *variabile-stringa*	Legge *variabile-stringa* da *nomestream*
char *nomestream.***get*()**	Legge un carattere da *nomestream*

Esempio:

```
string cognome;
ifstream filein ("amici.txt");   // apre filein in lettura
filein >> cognome;               // legge una stringa da filein
```

Operazioni sui file
Scrittura

L'operazione di **scrittura** su file si esegue come indicato:

nomestream << *variabile-stringa*	Scrive *nomestringa* su *nomestream*
*nomestream.***put** (**char** c)	Scrive *c* in *nomestream*

Esempio:
string cognome;
ofstream fileout ("testo.txt"); // apre *fileout* in scrittura
fileout << cognome; // scrive una stringa in *fileout*

Operazioni sui file
Chiusura

L'operazione di **chiusura** del file si esegue come indicato

*nomestream.***close**()	Chiude *nomestream* e rilascia il buffer

Esempio:
string cognome;
ofstream fileout ("testo.txt"); // apre *fileout* in scrittura
fileout << cognome; // scrive una stringa in *fileout*
fileout.**close** (); // chiude *fileout*

Operazioni sui file
Test su file

L'operazione di **test di fine file** si esegue come indicato

if (!*nomestream*) …. **if** (*nomestream*.**eof**()) …	Testa la fine del file *nomestream*

L'operazione di **test dell'esistenza del file** può essere espressa con

if (*nomestream*.**fail**()) ….	Testa l'esistenza del file *nomestream*

I file di testo

Vediamo ora alcuni esempi di codifica di programmi che fanno uso di file testo.

- Quando si scrive su un file testo, il testo è poi *leggibile mediante un qualunque editor* (es. Blocco Note di **Windows**)
- Quando si legge da un file testo, è possibile predisporre il *testo da leggere mediante un qualunque editor* (es. Blocco Note di **Windows**)

Lettura da file di testo

```
#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;
int main (int argc, char *argv[])
{   string nome;
    ifstream fin ("amici.dat");
    if (fin.fail ())
      cout<<"Errore: file inesistente!"<<endl;
    else
      while (fin>>nome)
         cout<<nome<<endl;
    fin.close();
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Questa istruzione **apre in lettura** il file sul disco avente nome esterno "amici.dat". Il file è gestito dal programma mediante l'oggetto file *fin*.

Il ciclo legge i nomi dal file e li stampa a video fino a quando *fin* risulta falso (raggiunta la fine del file)

Scrittura su file di testo

```
#include
….
int main (int argc, char *argv[])
{   string nome;
    char risposta='S';
    ifstream fout ("amici.dat", ios::in);
    if (!fout.fail())
    { cout<<"file esistente: sovrascrivere ? [S/N]";  cin>>risposta;    }
    if (toupper(risposta)=='S')
       caricamento dati in sovrascrittura;
    else
       caricamento dati in accodamento;
    fout.close();
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Quando si apre in scrittura un file è opportuno

- *testarne l'esistenza aprendolo in lettura*
- *chiedere conferma* all'utente se si vuole effettivamente *sovrascrivere*.

Scrittura su file di testo

```
…
if (!fout.fail())
{  cout<<"file esistente: sovrascrivere ? [S/N]"; cin>>risposta;    }
if (toupper (risposta)=='S')
{   ofstream fout ("amici.dat", ios::out);
    while (cin>>nome)       fout<<nome<<endl;
}
else
{  ofstream fout ("amici.dat", ios::app);
   while (cin>>nome)        fout<<nome<<endl;
}
fout.close();
system("PAUSE");
return EXIT_SUCCESS;
}
```


Accodamento su file di testo

Questo programma **apre in lettura** il file e ne controlla l'esistenza.

```
…
int main (int argc, char *argv[])
{   string nome;
    ifstream fout ("amici.dat", ios::in);
    if (fout.fail())
    { ofstream fout ("amici.dat", ios::out);     // riscrive
      while (cin>>nome)      fout<<nome<<endl;
    }
   else
   {  ofstream fout ("amici.dat", ios::app);   // accoda
      while (cin>>nome) fout<<nome<<endl;
   }
   fout.close();
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

La clausola **ios** può avere i seguenti valori:

- **in** – apertura in lettura
- **out** – apertura in scrittura
- **app** – apertura in accodamento

File da linea di comando

Lettura

```
#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;
int main(int argc, char *argv[])
{  char nome[20];
   if (argc<2)
   {  cout<<"Nome file: ";
      cin>>nome;
   }
   else strcpy (nome, argv[1]);
   ifstream fin (nome);
   elaborazione del file;
   fin.close();
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

Spesso è utile passare il nome del file da linea di comando mediante l'array di stringhe **argv[].**

Se **argc** < 2 manca il nome del file e quindi viene chiesto da input

File da linea di comando

Lettura

L'esecuzione si può ottenere:

- da linea di comando. Se il programma si chiama *Lettura.exe* e il file di dati *testo*, si può eseguire scrivendo:
  C:>Lettura testo
- dal Dev-Cpp impostando **Execute → Parameters** con il nome del file testo da elaborare.

File da linea di comando

Scrittura

```
#include …
using namespace std;
int main (int argc, char *argv[])
{  char nome[20];
   if (argc<2)
   {  cout<<"Nome file: ";
      cin>>nome;
   }
   else    strcpy (nome, argv[1]);
   ofstream fout (nome);
   while (cin>>nome)       fout<<nome<<endl;
   fout.close();
    system("PAUSE");
   return EXIT_SUCCESS;
}
```

In questo esempio il file viene scritto senza alcun controllo sulla sua esistenza.

Esempi

Gestire un elenco di nominativi

Vogliamo scrivere un unico programma a menu che consenta di gestire un elenco di nomi, tramite le seguenti operazioni utente:

- Creazione file
- Aggiunta nomi
- Stampa elenco nomi

Esempi
Gestire un elenco di nominativi

Librerie necessarie:

- **#include <cstdlib>**
- **#include <iostream>**
- **#include <fstream>**

Elenco prototipi

```
void scrittura();
void stampa();
void accoda();
void menu();
void esegui(string s);
```


Esempi
Gestire un elenco di nominativi

Il programma **main**() appare come:

```
int main (int argc, char *argv[])
{  string scelta;
   do
   {
    menu();
    cin>>scelta;
    esegui(scelta);
   } while (toupper (scelta[0])!='F');
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

La funzione *menu*() è la seguente:

```
void menu()
{
   cout<<"(C)rea"<<endl;
   cout<<"(A)ggiungi"<<endl;
   cout<<"(S)tampa"<<endl;
   cout<<"(F)ine"<<endl;
   cout<<"-->";
}
```

Esempi
Gestire un elenco di nominativi

La funzione *esegui* (**string** s) appare come:

```
void esegui (string s)
{   switch (toupper (s[0]))
    { case 'C':
            {   scrittura();   break;          }
      case 'A':
            {  accoda();      break;          }
      case 'S':
            {  stampa();      break;          }
      case 'F':     break;
      default:      cout<<"Scelta non ammessa!"<<endl;
    }
}
```


Esempi
Gestire un elenco di nominativi

La funzione *scrittura*() è:

```
void scrittura()
{
  ofstream fout (filename);
  string nome;
  cout<<"Nome: ";
  cin>>nome;
  cin.ignore(80,'\n');
  fout<<nome<<endl;
  fout.close();
}
```

La funzione *accoda*() è la seguente:

```
void accoda()
{
   ofstream fout( filename, ios::app);
   string nome;
   cout<<"Nome: ";
   cin>>nome;
   cin.ignore(80,'\n');
   fout<<nome<<endl;
   fout.close();
}
```

filename indica un array di caratteri contenente il nome esterno del file

Esempi
Gestire un file numerico

Sebbene i file testo *siano più lenti dei file binari* nell'elaborazione dei dati, è possibile utilizzarli anche per operazioni numeriche.

Vogliamo scrivere un programma che consenta di gestire una serie di valori numerici tramite le seguenti operazioni utente:

- Inserimento valori (con eventuale accodamento)
- Stampa valori
- Stampa della somma dei valori

Esempi
Gestire un file numerico

Librerie e definizioni:

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#define filename "Numeri.txt"
```

Elenco prototipi:

```
void inserisci (char fname[12], int n);
void stampa (char fname[12]);
int somma (char fname[12]);
```

Esempi
Gestire un file numerico

Il programma **main**() appare come:

```
int main (int argc, char *argv[])
{
   int num;
   while (cin>>num)
     inserisci (filename, num);
   stampa (filename);
   cout<<"Somma: "<<somma (filename)<<endl;
   system("PAUSE");
   return EXIT_SUCCESS;
}
```


Esempi
Gestire un file numerico

La funzione *inserisci*(…) appare come:

```
void inserisci (char fname[12], int n)
{  ifstream f(filename, ios::in);
   if (f.eof())                    // Il file non esiste, apre in scrittura...
   {       ofstream f(filename, ios::out);
           f<<n<<endl;
   }
   else                            // Il file esiste, apre in accodamento
   {       ofstream f(filename, ios::app);
           f<<n<<endl;
   }
   f.close();
}
```

Esempi
Gestire un file numerico

La funzione *stampa*(…) appare come:

```
void stampa (char fname[12])
{
    int num;
    fstream f(fname, ios::in);
    if (!f)
     cout<<"File inesistente"<<endl;
    else
    {
      while (f>>num)
         cout<<num<<endl;
    }
    f.close();
}
```


Esempi
Gestire un file numerico

La funzione *somma*(…) appare come:

```
int somma (char fname[12])
{
   int s=0, num;
   fstream f(fname, ios::in);
   if (!f)
     cout<<"File inesistente"<<endl;
   else
   {
      while (f>>num)
         s+=num;
    }
   f.close();
   return s;
}
```

Cosa ho imparato

- Classi di input/output
- File testo in C++
- Operazioni su file testo

Cosa ho imparato a fare

- Scrivere applicazioni per leggere da file testo
- Scrivere applicazioni per registrare dati su file testo
- Usare file per riscrivere o per accodare dati
- Scrivere una semplice applicazione per gestire un file

Terminologia

- Stream di input
- Stream di output
- Stream di input/output
- **ifstream**
- **ofstream**
- **fstream**
- **ios::in**
- **ios::out**
- **ios::app**
- **fail()**
- **open()**
- **close()**
- **eof()**

Altre fonti di informazione

- J. Purdum, C - ed. Jackson
- Romagnoli Ventura, C/C++ - Ed. Petrini
- A. Lorenzi et alii – Il linguaggio C++ - Ed. ATLAS
- A. Garavaglia, F.Petracchi, S.Forte
  Strutture dati e programmazione per oggetti, ed. Masson Scuola