L'argomento della lezione

I FILE

Strutture dati

in memoria secondaria

Dall'informazione al file

Per permettere il riutilizzo di informazioni oltre la durata dei programmi che le hanno prodotte, i dati sono allocati su dispositivi in memoria secondaria.

- **I BIT:** sono l'unità elementare d'informazione [es: 0,1]
- **I CARATTERI:** utilizzano sequenze di bit per rappresentare un singolo numero, lettera o simbolo speciale [es: 8,9… a,b… /,&,%...]
- **I CAMPI:** sono gruppi di caratteri organizzati sintatticamente per trasportare un'informazione [es: nome]
- **I RECORD:** sono composti da diversi campi semanticamente pertinenti fra loro, rappresentati mediante una classe C++ [es: persona (nome, cognome, …]

I file: gruppi di record correlati fra loro

Es: file registro studenti ASD

(NB: il nome "*File fisico"* indica invece un insieme di blocchi, intesi come insiemi di byte e connessi al concetto di settore fisico**)**

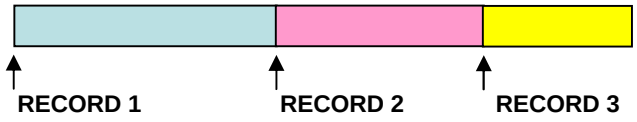
La struttura dei file

Organizzazione logica dei file

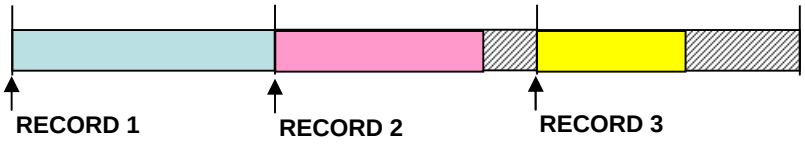
•**LA CHIAVE DEL RECORD:** è particolare campo che permette l'individuazione univoca di un record all'interno di un file

Sulla base della chiave del record i file possono permettere due modalità di accesso:

SEQUENZIALE:
I record sono semplicemente inseriti uno di seguito all'altro sulla base della chiave progressiva



CASUALE:
I record hanno lunghezza fissa ed è possibile accedere ad uno particolare di essi senza scandirli tutti



L'inserimento di dati in un file ad accesso casuale non comporta la distruzione di quelli immagazzinati, ma si deve far fronte al problema del dimensionamento dei settori e della minimizzazione degli sprechi.

L'inserimento di dati in un file ad accesso sequenziale deve necessariamente avvenire secondo l'ordine progressivo della chiave

La struttura dei file

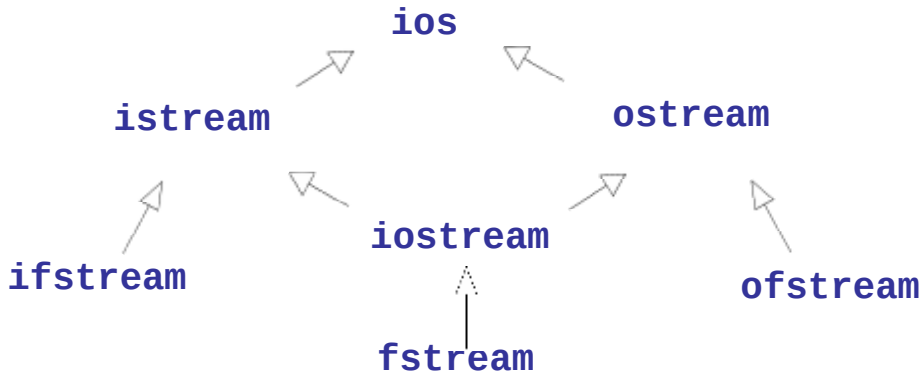
I file nel linguaggio C++

- **Sono visti come uno stream di byte conclusi da un marcatore "end-of-file"**
- **Richiedono l'uso delle classi:**
  - per la gestione degli stream (`#include <iostream.h>`)
  - per la gestione degli input da file (`#include <ifstream.h>`)
  - per la gestione dell'output su file (`#include <ofstream.h>`)
  - per la gestione di input e output da/su file (`#include <fstream.h>`)

La struttura dei file

In particolare:

I metodi di queste classi saranno ereditati dalle classi per la gestione degli stream generici di input e output



Modalità apertura	Descrizione
`ios::app`	Aggiunge l'output alla fine del file
`ios::ate`	Apre un file e si sposta alla fine di esso
`ios::in`	Apre un file in input
`ios::out`	Apre un file in output
`ios::trunc`	Elimina il contenuto del file (se esiste)
`ios::nocreate`	Interrompe l'operazione open se il file non esiste
`ios::noreplace`	Interrompe l'operazione open se il file esiste

Creazione di un file sequenziale

```
#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;

#include <fstream>

using std::ofstream;
```

Equivale a using namespace std … std::cout

Serve a evitare conflitti individuando il namespace del metodo invocato nel caso di overloading di operatori

Enter the account, name, and balance.
Enter end-of-file to end input.
? 100 Rossi 23280
? 200 Bianchi 11290
? ^Z

Creazione di un file sequenziale

Crea un oggetto *outClientFile* e lo associa al file fisico *clients.dat*

```
int main()
{
   // ofstream constructor opens file
   ofstream outClientFile( "clients.dat", ios::out );

   if ( !outClientFile ) {  // overloaded ! operator
      cerr << "File could not be opened" << endl;
      exit( 1 );
   }

   cout << "Enter the account, name, and balance.\n"
        << "Enter end-of-file to end input.\n? ";

   int account;
   char name[ 30 ];
   double balance;

   while ( cin >> account >> name >> balance ) {
      outClientFile << account << ' ' << name
                    << ' ' << balance << '\n';
      cout << "? ";
   }

   return 0;  // ofstream destructor closes file
}
```

Read&print di file sequenziale

```
#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;

#include <fstream>

using std::ifstream;

#include <iomanip>

using std::setiosflags;
using std::resetiosflags;
using std::setw;
using std::setprecision;
```

Lettura dei record presenti in un file e loro visualizzazione tabulata su monitor

Account	Name	Balance
100	Rossi	23280
200	Bianchi	11290

Dichiarazione anticipata

```
void outputLine( int, const char *, double );
```

Read&print di file sequenziale

Gestione dell'errore in apertura

Tabula la visualizzazione

Legge dal file finché non trova l'*End-Of-File*

```
int main()
{
   // ifstream constructor opens the file
   ifstream inClientFile( "clients.dat", ios::in );

   if ( !inClientFile ) {
      cerr << "File could not be opened\n";
      exit( 1 );
   }

   int account;
   char name[ 30 ];
   double balance;

   cout << setiosflags( ios::left ) << setw( 10 ) <<
"Account"
        << setw( 13 ) << "Name" << "Balance\n"
        << setiosflags( ios::fixed );

   while ( inClientFile >> account >> name >> balance )
      outputLine( account, name, balance );

   return 0;  // ifstream destructor closes the file
}
```

Read&print di file sequenziale

Il metodo *outputLine* particolarizza l'operatore *cout* predefinendo una tabulazione per la visualizzazione di una tripla di parametri

Account	Name	Balance
100	Rossi	23280
200	Bianchi	11290

```
void outputLine( int acct, const char *name, double bal )
{
   cout << setiosflags( ios::left ) << setw( 10 ) << acct
        << setw( 13 ) << name << setw( 7 ) << setprecision( 2 )
        << resetiosflags( ios::left )
        << bal << '\n';
}
```

Creazione di un file random

Creazione di sistema di elaborazione dei crediti capace di memorizzare 100 record a lunghezza fissa

```
#include <iostream>

using std::cerr;
using std::endl;
using std::ios;

#include <fstream>

using std::ofstream;
```

Permette di utilizzare i costrutti di struttura di un record Cliente

```
#include "clntdata.h"
```

```
//clntdata.h definizione della struct clientData
#ifndef CLNTDATA_H
#define CLNTDATA_H
struct clientData {
        int accountNumber;
        char lastName [15];
        char firstName [10];
        float balance;
};
#endif
```

NB: la struct non riserva spazio in memoria

Creazione di un file random

```
int main()
{
   ofstream outCredit( "credit.dat", ios::out );

   if ( !outCredit ) {
      cerr << "File could not be opened." << endl;
      exit( 1 );
   }

   clientData blankClient = { 0, "", "", 0.0 };

   for ( int i = 0; i < 100; i++ )
      outCredit.write(
         reinterpret_cast<const char *>( &blankClient ),
         sizeof( clientData ) );
   return 0;
}
```

Dichiarazione del file e assegnazione al file fisico

Gestione dell'errore in apertura

Crea un oggetto *ClientData*

Effettua la scrittura su file, uniformando lo spazio per *blankClient* al valore previsto per *clientData*

Converte l'indirizzo di *blankClient* in un puntatore *const char** e fissa il numero di byte in scrittura

Scrittura in un file random

Inserimento non sequenziale di record in un file random basandosi sul campo chiave

```
Enter account number (1 to 100)
? 15
Enter lastname, firstname, balance
? Lippi Marcello 95000000
Enter account number (1 to 100)
…
```

```
#include <iostream>

using std::cerr;
using std::endl;
using std::cout;
using std::cin;
using std::ios;

#include <fstream>

using std::ofstream;

#include <cstdlib>
#include "clntdata.h"

int main()
{
   ofstream outCredit( "credit.dat", ios::out );

   if ( !outCredit ) {
      cerr << "File could not be opened." << endl;
      exit( 1 );
   }

   cout << "Enter account number "
        << "(1 to 100)\n? ";

   clientData client;
   cin >> client.accountNumber;

                              …
```

Crea il record da inserire nel file

Legge il primo campo del record

Scrittura in un file random

Verifica che il campo chiave appartenga all'intervallo ammissibile

In caso affermativo richiede la digitazione degli altri campi e li legge

Imposta il puntatore di "put" in un preciso blocco fisico

```
while ( client.accountNumber > 0 &&
        client.accountNumber <= 100 ) {
   cout << "Enter lastname, firstname, balance\n? ";
   cin >> client.lastName >> client.firstName
       >> client.balance;

   outCredit.seekp( ( client.accountNumber - 1 ) *
                    sizeof( clientData ) );
   outCredit.write(
      reinterpret_cast<const char *>( &client ),
      sizeof( clientData ) );

   cout << "Enter account number\n? ";
   cin >> client.accountNumber;
}

return 0;
}
```

Domanda: come uscireste dal ciclo

??

Lettura sequenziale da file random

```
#include <iostream>

using std::cout;
using std::endl;
using std::ios;
using std::cerr;

#include <iomanip>

using std::setprecision;
using std::setiosflags;
using std::resetiosflags;
using std::setw;

#include <fstream>

using std::ifstream;
using std::ostream;

#include <cstdlib>
#include "clntdata.h"

void outputLine( ostream&, const clientData & );
```

Account	Last Name	FirstName	Balance
18	Mancini	Roberto	23280
21	Ancelotti	Carlo	21290
23	Capello	Fabio	24020
24	Caso	Mimmo	-5000

Lettura sequenziale da file random

```
int main()
{
   ifstream inCredit( "credit.dat", ios::in );

   if ( !inCredit ) {
      cerr << "File could not be opened." << endl;
      exit( 1 );
   }

   cout << setiosflags( ios::left ) << setw( 10 ) << "Account"
        << setw( 16 ) << "Last Name" << setw( 11 )
        << "First Name" << resetiosflags( ios::left )
        << setw( 10 ) << "Balance" << endl;

   clientData client;

   inCredit.read( reinterpret_cast<char *>( &client ),
                  sizeof( clientData ) );

   while ( inCredit && !inCredit.eof() ) {

      if ( client.accountNumber != 0 )
         outputLine( cout, client );

      inCredit.read( reinterpret_cast<char *>( &client ),
                     sizeof( clientData ) );
   }

   return 0;
}
```

Legge il primo record del file, assegnandogli una dimensione standard

Domanda: cosa succede in questo ciclo ??

Dai file alle liste

Lettura di tutti i record di un file e inserimento in una lista

```
…
ifstream inCredit( "credit.dat", ios::in );
ListaT<clientData> L;
clientData client;
…
inCredit.read( reinterpret_cast<char *>( &client ),
                    sizeof( clientData ) );
while ( inCredit && !inCredit.eof() ) {

      if ( client.accountNumber != 0 )
         L.append( client );

      inCredit.read( reinterpret_cast<char *>( &client ),
                     sizeof( clientData ) );
   }
…
```

Dai file ai vettori

Lettura di tutti i record di un file e inserimento in un vettore

```
…
ifstream inCredit( "credit.dat", ios::in );
…
inCredit.read( reinterpret_cast<char *>( &client ),
                    sizeof( clientData ) );
int i=0;
while ( !inCredit.eof() ) { //trascurando l'errore in lettura

       inCredit.read( reinterpret_cast<char *>( &client ),
                         sizeof( clientData ) );
       i++;
}

ifstream inCredit( "credit.dat", ios::in );
clientData V[i];
clientData client;
i=0;
inCredit.read( reinterpret_cast<char *>( &client ),
                    sizeof( clientData ) );
while ( !inCredit.eof() ) {

      if ( client.accountNumber != 0 )
         V[i]=client;

      inCredit.read( reinterpret_cast<char *>( &client ),
                         sizeof( clientData ) );
      i++;
}
…
```