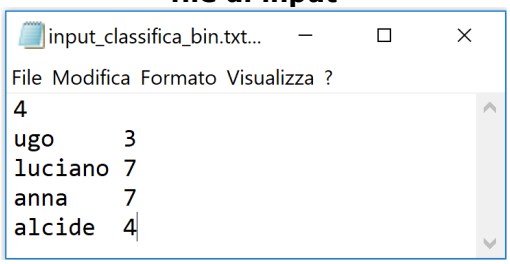
Programma di **ordinamento del file atleti per punteggio** decrescente **e nome** crescente utilizzando la funzione ***sort*** della libreria standard del C++ (**algorithm**) sulla Tabella **atleti** (array di struttura **Atleta**, in *memoria centrale*) e salvando successivamente la Tabella **atleti** su *memoria di massa* in un **File** di tipo **binary**.

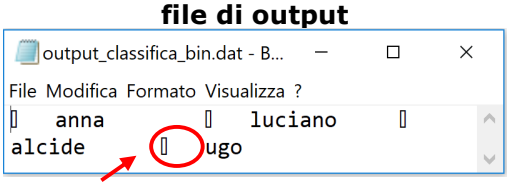
```
#include <iostream>
using namespace std;
#include <string.h>
#include <fstream>
#include <algorithm>
#define DIM 12
struct Atleta {
 int punti;
 char NOME[DIM];
};
bool confronta(const Atleta& uno, const Atleta& due){
      if (uno.punti > due.punti)
            return true;
      else
      if (uno.punti == due.punti
      && strcmp (uno.NOME, due.NOME) < 0)
            return true;
    else
        return false;
}
void scritturaFile (Atleta atleti[], int N){
  ofstream fout("output_classifica_bin.dat",
                ios::binary);
  for (int i=0; i<N; i++)
    fout.write( (char *) &atleti[i], sizeof(Atleta) );
  fout.close();
}
void letturaFile (Atleta atleti[], int& N){
  ifstream finp("input_classifica_bin.txt");
    finp>> N;
    for (int i=0; i<N; i++){
        for (int k=0; k<DIM; k++)
            atleti[i].NOME[k] = 0x20;
        finp >> atleti[i].NOME >> atleti[i].punti;
    }
    finp.close();
}
int main(){
  int N, i;
  Atleta atleti[100];
    letturaFile  (atleti, N);
    sort (atleti, atleti+N, confronta);
    scritturaFile(atleti, N);
}
```

file di input



In questa versione del programma il file di output è di tipo **binary** quindi con Blocco Note sono leggibili solo i nomi (stringhe di 12 char); il punteggio è un intero (in formato binario) che occupa 4 byte esposto da Blocco Note come

file di output

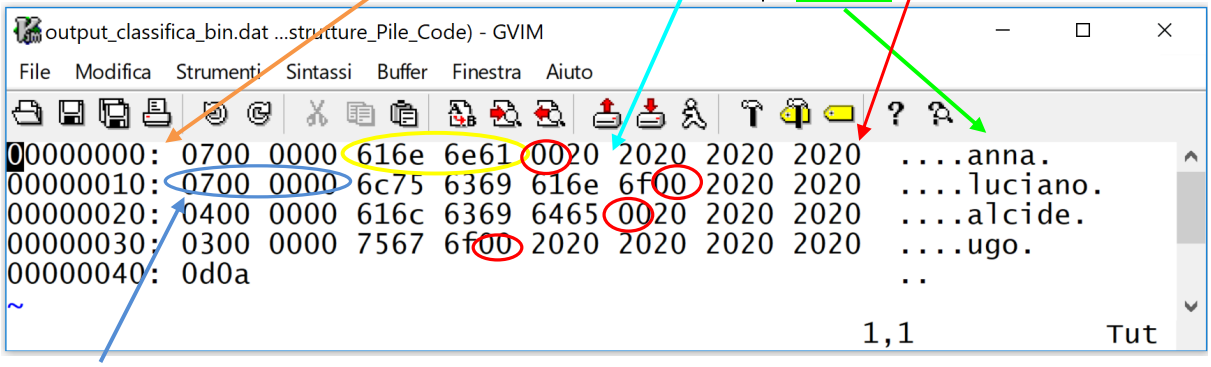


un rettangolo seguito apparentemente da spazi.

ciclo per inizializzare a SPAZI tutti i caratteri dell'elemento atleti[i].NOME

Il file di output ha record di **lunghezza costante da 16 byte** (4+12), visualizzati in basso mediante l'editor ***VIM*** *(Vi IMproved)* utilizzando *Strumenti>Converti a esadecimale*.

L'editor espone :
- a sinistra il progressivo in esadecimale del primo byte di riga (0 per la prima, $10_{16} = 16_{10}$ per la seconda, etc);
- al centro il contenuto di 8 coppie di byte del file convertiti in esadecimale (esempio: `0700 0000` sono i 7 punti di anna, `616e 6e61` la codifica ASCII dei caratteri **an na** seguiti dal fine stringa `00`, gli altri 7 byte a spazi , `20` è il codice ASCII per spazio)
- a destra il contenuto come in Blocco Note



Nei primi 4 byte di ogni record è registrato il punteggio (`atleti[i].punti`) secondo l'ordine **little-endian** (memorizzazione che inizia dal byte meno significativo per finire con il più significativo) utilizzato dai processori Intel.

```
#include <iostream>
using namespace std;
#include <fstream>
#include <string.h>
#include <iomanip>

#define LN 24

struct Lavoratore {
 unsigned int    ID;         //-- 4 byte
 char            nome[LN];
 unsigned int    stipendio; //-- 4 byte
};

void spazi_nel_nome(char testo[]){
    for (int i=0; i< LN; i++)
        testo[i] = ' ';       //-- SPAZIO
}

int main(){
  Lavoratore dipendente;
  ofstream fout;
    fout.open("anagrafe_bin.dat", ios::app | ios::binary);
    if (fout.fail())
        cout << "file non esiste\n";

    cout<< "matricola dipendente (0=fine): ";
    cin>> dipendente.ID;
    while (dipendente.ID != 0) {
          cin.ignore(80, '\n');
          cout<< "cognome e nome dipendente    : ";

   //---- spazi_nel_nome(dipendente.nome);

          cin.getline(dipendente.nome, LN);
          cout<< "stipendio                    : ";
          cin>> dipendente.stipendio;
          fout.write( (char *) &dipendente, sizeof(dipendente) );
          cout<< "\nmatricola dipendente (0=fine): ";
          cin>> dipendente.ID;
        }
    fout.close();

  ifstream finp;
    finp.open("anagrafe_bin.dat", ios::in | ios::binary);
    if ( finp.fail() )
         cout<<"\n Errore nell'apertura dell'archivio\n";
    else {
         cout<<"\n Elenco dipendenti registrati\n";
         while (finp.read( (char *) &dipendente, sizeof(dipendente) ))  {
               cout<<setw(5)          << dipendente.ID         << '\t';
               cout<<left  <<setw(LN)<< dipendente.nome       << '\t';
               cout<<right <<setw(10)<< dipendente.stipendio << '\n';
         }
         cout<<endl;
         finp.close();
    }
    return 0;
}
```

SECONDO ESEMPIO p322es6_bin_int.cpp

Programma per la gestione delle anagrafiche dei dipendenti di una azienda (libro di testo p.322).

Il programma acquisisce da tastiera i dati dei dipendenti (`ID`, intero senza segno per la matricola; `nome`, 24 caratteri per il cognome e nome, `stipendio`) utilizzando la variabile strutturata `dipendente` di tipo `Lavoratore` (struttura dati definita da programma); accoda quindi i dati al file "anagrafe_bin.dat" aperto in append (`ios::app`) come un unico record da 32 byte (4+24+4) mediante il metodo `write` dell'oggetto `fout` di tipo `ofstream`.

Conclusa l'acquisizione (digitando 0 per una nuova matricola), chiude il file aperto in *output* (nome logico `fout`) e lo riapre in *input* (`finp`) per esporre a video tutti i dipendenti.

Se il file non esiste (`fout.fail()` risulta vera) viene data segnalazione a video e l'elaborazione continua perchè il file viene creato con `fout.open()` .

La lettura da testiera mediante cin termina con ***spazio***, ***tabulazione*** o ***invio*** ma lascia il carattere `\n` nel buffer, carattere che va scartato dalla successiva acquisizione del `nome` la cui lettura deve, quindi, essere preceduta dal metodo: `cin.ignore(80, '\n')`.

La funzione **spazi_nel_nome()** è un commento, quindi NON viene eseguita.

La lettura da testiera dell'elemento `dipendente.nome` della variabile `dipendente` mediante cin.getline consente l'acquisizione anche dello spazio che separa il cognome dal nome perchè questo metodo acquisisce i caratteri digitati fino all'***invio*** (o, al massimo, LN-1 caratteri) aggiungendo in coda (oppure in posizione LN-1) il terminatore di stringa (`\0`).

Dopo l'acquisizione il programma chiude il file aperto in *output* (nome logico `fout`) e lo riapre in input (`finp`) per esporre a video tutti i dipendenti.

Il metodo setw(x) della libreria **iomanip** del C++ imposta la larghezza (*width*) del campo che segue;

left allinea l'output a sinistra, `'\t'` serve a tabulare,

right allinea l'output a destra.

aspetto della videata per l' ESECUZIONE del programma p322es6_bin_int

```
matricola dipendente (0=fine): 1
cognome e nome dipendente    : Baldi Luigi
stipendio                    : 255

matricola dipendente (0=fine): 2
cognome e nome dipendente    : Carli Franco
stipendio                    : 65535

matricola dipendente (0=fine): 3
cognome e nome dipendente    : Franchi Anna Maria
stipendio                    : 16777215

matricola dipendente (0=fine): 257
cognome e nome dipendente    : Sesti Carlo
stipendio                    : 4294967295

matricola dipendente (0=fine): 0

 Elenco dipendenti registrati
   1   Baldi Luigi                  255
   2   Carli Franco               65535
   3   Franchi Anna Maria      16777215
 257   Sesti Carlo           4294967295
```

Inserendo i dati di 4 dipendenti, vengono inseriti nel file i valori esadecimali (utilizzando l'ordine **little-endian**):

1		01 00 00 00
FF	(255)	FF 00 00 00
2		02 00 00 00
FFFF	(65535)	FF FF 00 00
3		03 00 00 00
FFFFFF	(16777215)	FF FF FF 00
FF		FF 00 00 00
FFFFFF FF		FF FF FF FF

(4.294.967.295 massimo valore per int)

← dati esposti a video (leggendo finp)

File *anagrafe_bin.dat* visto mediante l'editor *VIM* e utilizzando *Strumenti* > *Converti a esadecimale*

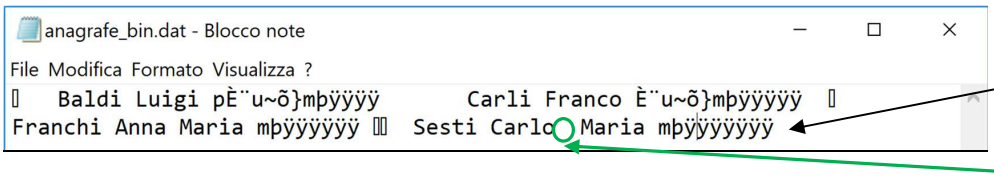
NOTA: ogni nome è seguito dal fine stringa (in esadecimale **00** è la codifica del **NULL character**)

File Modifica Strumenti Sintassi Buffer Finestra Aiuto

```
00000000: 0100 0000 4261 6c64 6920 4c75 6967 6900  ....Baldi Luigi.
00000010: 70c8 a875 7ef5 7d6d feff ffff ff00 0000  p..u~.}m........
00000020: 0200 0000 4361 726c 6920 4672 616e 636f  ....Carli Franco
00000030: 00c8 a875 7ef5 7d6d feff ffff ffff 0000  ...u~.}m........
00000040: 0300 0000 4672 616e 6368 6920 416e 6e61  ....Franchi Anna
00000050: 204d 6172 6961 006d feff ffff ffff ff00   Maria.m........
00000060: 0101 0000 5365 7374 6920 4361 726c 6f00  ....Sesti Carlo.
00000070: 204d 6172 6961 006d feff ffff ffff ffff   Maria.m........
00000080: 0d0a                                     ..
~
```

NOTA: **0d0a** rappresenta il fine linea e, in questo caso, anche il fine file - ogni **.** un carattere non stampabile
0A è la codifica del carattere ASCII **new line** LF (*Line Feed)* ovvero del carattere di escape del C++ **\n**
0D è la codifica del carattere ASCII **carriage return** CR (posizionamento ad inizio riga).

File *anagrafe_bin.dat* visto mediante *Blocco Note*



l'elemento `dipendente.nome` non è inizializzato quindi contiene valori casuali che NON vengono sostituiti dalla lettura che imposta solo fino al **NULL**

File *anagrafe_bin.dat* ottenuto utilizzando la funzione **spazi_nel_nome(dipendente.nome)**

```
00000000: 0100 0000 4261 6c64 6920 4c75 6967 6900  ....Baldi Luigi.
00000010: 2020 2020 2020 2020 2020 2020 ff00 0000                ....
00000020: 0200 0000 4361 726c 6920 4672 616e 636f  ....Carli Franco
00000030: 0020 2020 2020 2020 2020 2020 ffff 0000  .            ....
```